

PlayStation®2 EE Library Reference Release 2.7

Graphics Libraries

© 2003 Sony Computer Entertainment Inc.

Publication date: March 2003

Sony Computer Entertainment Inc.
1-1, Akasaka 7-chome, Minato-ku
Tokyo 107-0052, Japan

Sony Computer Entertainment America
919 E. Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

The *PlayStation®2 EE Library Reference - Graphics Libraries* manual is supplied pursuant to and subject to the terms of the Sony Computer Entertainment PlayStation® license agreements.

The *PlayStation®2 EE Library Reference - Graphics Libraries* manual is intended for distribution to and use by only Sony Computer Entertainment licensed Developers and Publishers in accordance with the PlayStation® license agreements.

Unauthorized reproduction, distribution, lending, rental or disclosure to any third party, in whole or in part, of this book is expressly prohibited by law and by the terms of the Sony Computer Entertainment PlayStation® license agreements.

Ownership of the physical property of the book is retained by and reserved by Sony Computer Entertainment. Alteration to or deletion, in whole or in part, of the book, its presentation, or its contents is prohibited.

The information in the *PlayStation®2 EE Library Reference - Graphics Libraries* manual is subject to change without notice. The content of this book is Confidential Information of Sony Computer Entertainment.

 and PlayStation are registered trademarks of Sony Computer Entertainment Inc. All other trademarks are property of their respective owners and/or their licensors.

Summary Table of Contents

About This Manual	v
Changes Since Last Release	v
Related Documentation	viii
Typographic Conventions	viii
Developer Support	viii
Chapter 1: Basic Graphics Library	1-1
Chain Management Structure	1-5
Register Unions	1-6
Packet Structures	1-8
Parameter Structure	1-18
Utility Functions (Some Are Macro Functions)	1-19
Chain Manipulation Functions	1-24
Packet Control Functions	1-42
Packet Default Value Setting Functions	1-44
Packet Initialization Functions	1-50
Set Functions (For Non-Drawing Packets)	1-61
Index Acquisition Functions (For Drawing Packets)	1-80
Set Functions (For Drawing Packets, Some Are Macro Functions)	1-94
Individual Packet Transfer Functions	1-109
Chapter 2: GS Basic Library	2-1
Structures	2-3
Functions	2-16
Chapter 3: DMA Packet Management Services	3-1
Functions	3-3
Chapter 4: High Level Graphics Library	4-1
Structures	4-3
Functions	4-12
Chapter 5: GS Register Management Services	5-1
Common Structures	5-5
Display Environment Setting Structures	5-6
GS Local Memory Management Structures	5-7
Context Management Structures	5-8
Old Structures	5-13
Display Environment Setting Functions	5-17
GS Local Memory Management Functions	5-23
GS Register Setting Functions	5-33
Old GS Register Setting Functions	5-110
Chapter 6: Memory Area Management Services	6-1
Functions	6-3
Chapter 7: High Level Graphics Plugin Library	7-1
Structures	7-5
Functions	7-45

Chapter 8: VU0 Matrix Arithmetic Library	8-1
Structures	8-3
Functions	8-6

About This Manual

This is the Runtime Library Release 2.7 version of the *PlayStation®2 EE Library Reference - Graphics Libraries* manual.

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

Changes Since Last Release

The following list is a compilation of all changes since Runtime Library Release 2.5.7.

Chapter 2: GS Basic Library

- In the "Calling Conditions" of the following functions, the description "Cannot be called from an interrupt handler" was changed to "Can be called from an interrupt handler".

- `sceGsExecLoadImage()`
 - `sceGsGetGParam()`
 - `sceGsPutDispEnv()`
 - `sceGsPutDrawEnv()`
 - `sceGsResetPath()`
 - `sceGsSetDefAlphaEnv()`
 - `sceGsSetDefAlphaEnv2()`
 - `sceGsSetDefClear()`
 - `sceGsSetDefClear2()`
 - `sceGsSetDefDrawEnv()`
 - `sceGsSetDefDrawEnv2()`
 - `sceGsSetDefLoadImage()`
 - `sceGsSetDefStoreImage()`
 - `sceGsSetDefTexEnv()`
 - `sceGsSetDefTexEnv2()`
 - `sceGsSetHalfOffset()`
 - `sceGsSetHalfOffset2()`
 - `sceGsSwapDBuff()`
 - `sceGsSwapDBuffDc()`
 - `sceGsSyncPath()`

- A description of the calling conditions for `isceGsGetIMR()` was added.
- A description of the calling conditions for `isceGsPutIMR()` was added.
- In "Calling Conditions" of the following functions, the description "Not multithread safe" was changed to "Multithread safe".

- `sceGsSetDefAlphaEnv()`
 - `sceGsSetDefAlphaEnv2()`
 - `sceGsSetDefClear()`
 - `sceGsSetDefClear2()`

sceGsSetDefDrawEnv()
 sceGsSetDefDrawEnv2()
 sceGsSetDefLoadImage()
 sceGsSetDefStoreImage()
 sceGsSetDefTexEnv()
 sceGsSetDefTexEnv2()
 sceGsSetHalfOffset()
 sceGsSetHalfOffset2()

- In "Calling Conditions" of the following functions, the description "does not depend on interrupt disabled or enabled state" was changed to "must be called in an interrupt enabled state".

sceGsSetDefDBuff()
 sceGsSetDefDBuffDc()
 sceGsSetDefDispEnv()
 sceGsSyncV()

Chapter 3: DMA Packet Management Services

- Descriptions of the following functions were added.

sceHiDMADeleteMarked()
 sceHiDMAGetChainHead()
 sceHiDMAGetChainTail()
 sceHiDMAMarkup()
 sceHiDMARegistClear()

- An explanation of the DMA Buffer was added to the "Description" of the following functions.

sceHiDMAInit_DBuf()
 sceHiDMAMake_CallID()
 sceHiDMAMake_CallPtr
 sceHiDMAMake_ContinueMicro()
 sceHiDMAMake_ExecMicro()
 sceHiDMAMake_ExecMicroAddr()
 sceHiDMAMake_LoadGS()
 sceHiDMAMake_LoadGSLump()
 sceHiDMAMake_LoadImm()
 sceHiDMAMake_LoadMicro()
 sceHiDMAMake_LoadPtr()
 sceHiDMAMake_LoadStep()
 sceHiDMAMake_Lump()
 sceHiDMAMake_LumpEnd()
 sceHiDMAMake_WaitMicro()
 sceHiDMARegist()

- The "Calling Conditions" of sceHiDMAPurge() were changed as follows:

Before: Not multithread safe
 After: Multithread safe

- A description of waiting for the end of transfer was added to the "Description" of `sceHiDMASend()`.
- The "Calling Conditions" of `sceHiDMASend()` were changed as follows:

Before:	Can be called from the thread
After:	Cannot be called from the thread

Chapter 4: High Level Graphics Library

- Descriptions of the following functions were added.

`sceHiCalcPlugBlkSize()`

`sceHiCreateDataBlk()`

`sceHiCreatePlugBlk()`

Chapter 7: High Level Graphics Plugin Library

- Descriptions of the following functions have been added.

<code>sceHiPlugHrchyGetMatrix()</code>
<code>sceHiPlugMatrix()</code>
<code>sceHiPlugManime()</code>
- Descriptions of interpolation type and function curve were added to the "Description" of the `sceHiPlugAnimeHead_t` structure.
- A description of the light color and light direction was added to the "Description" of the `sceHiPlugMicroLight_t` structure.
- A description of the `sceHiPlugHrchyMatrix_t` structure was added.
- In the "Return Value" of the following functions, the description of `SCE_HIG_INVALID_DATA` was changed to "The stack area is NULL".

`sceHiPlugAnime()`

`sceHiPlugCamera()`

`sceHiPlugClip()`

`sceHiPlugClutBump()`

`sceHiPlugFishEye()`

`sceHiPlugHrchy()`

`sceHiPlugLightMap()`

`sceHiPlugMicro()`

`sceHiPlugReflect()`

`sceHiPlugRefract()`

`sceHiPlugShadowBox()`

`sceHiPlugShadowMap()`

`sceHiPlugShape()`

`sceHiPlugShapeInvisible()`

`sceHiPlugShapeMasterChainSetting()`

`sceHiPlugShare()`

`sceHiPlugSkin()`

`sceHiPlugTex2D()`

```
sceHiPlugTim2()
sceHiPlugTim2SetPicture()
```

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
<code>courier</code>	Indicates literal program code.
<i>italic</i>	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
<i>In North America:</i>	<i>In North America:</i>
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: https://www.ps2-pro.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
<i>In Europe:</i>	<i>In Europe:</i>
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: Basic Graphics Library

Table of Contents

Chain Management Structure	1-5
sceGpChain	1-5
Register Unions	1-6
sceGpPack	1-6
sceGpReg	1-7
Packet Structures	1-8
sceGpAd	1-8
sceGpAlphaEnv	1-9
sceGpCall	1-10
sceGpLoadImage	1-11
sceGpLoadTexelClut	1-12
sceGpPrimP	1-13
sceGpPrimR	1-14
sceGpRef	1-15
sceGpTexEnv	1-16
sceGpTexEnvMipmap	1-17
Parameter Structure	1-18
sceGpTextureArg	1-18
Utility Functions (Some Are Macro Functions)	1-19
sceGpChkChainOtSize	1-19
sceGpChkNumPtoV	1-20
sceGpChkPacketSize	1-21
sceGpGetTail	1-22
sceGpGetTailChain	1-23
Chain Manipulation Functions	1-24
sceGpAddChain	1-24
sceGpAddChain2	1-25
sceGpAddPacket	1-26
sceGpAddPacket2	1-27
sceGpCallChain	1-28
sceGpCallChain2	1-29
sceGpEndLevel	1-30
sceGpInitChain	1-31
sceGpKickChain	1-32
sceGpKickChain2	1-33
sceGpPrintChain	1-34
sceGpRemoveNextChain	1-35
sceGpRemoveNextPacket	1-36
sceGpResetChain	1-37
sceGpResetChainRev	1-38
sceGpSearchTailToRemove	1-39
sceGpSetStartLevel	1-40
sceGpTermChain	1-41
Packet Control Functions	1-42
sceGpCopyPacket	1-42

sceGpInsertPacket	1-43
Packet Default Value Setting Functions	1-44
sceGpSetDefaultAa1	1-44
sceGpSetDefaultAbe	1-45
sceGpSetDefaultCtxt	1-46
sceGpSetDefaultDirectHL	1-47
sceGpSetDefaultFog	1-48
sceGpSetDefaultZ32	1-49
Packet Initialization Functions	1-50
sceGpInitAd	1-50
sceGpInitAlphaEnv	1-51
sceGpInitCall	1-52
sceGpInitLoadImage	1-53
sceGpInitLoadTexelClut	1-54
sceGpInitPacket	1-55
sceGpInitPrimP	1-56
sceGpInitPrimR	1-57
sceGpInitRef	1-58
sceGpInitTexEnv	1-59
sceGpInitTexEnvMipmap	1-60
Set Functions (For Non-Drawing Packets)	1-61
sceGpSetAd	1-61
sceGpSetAlphaEnv	1-62
sceGpSetAlphaEnvFunc	1-65
sceGpSetCall	1-68
sceGpSetLoadImage	1-69
sceGpSetLoadImageByArgTim2	1-70
sceGpSetLoadImageByTim2	1-71
sceGpSetLoadTexelClut	1-72
sceGpSetLoadTexelClutByArgTim2	1-73
sceGpSetLoadTexelClutByTim2	1-74
sceGpSetRef	1-75
sceGpSetTexEnv	1-76
sceGpSetTexEnvByArgTim2	1-77
sceGpSetTexEnvByDrawEnv	1-78
sceGpSetTexEnvByTim2	1-79
Index Acquisition Functions (For Drawing Packets)	1-80
sceGpIndexQ#_R	1-80
sceGpIndexQ_R	1-81
sceGpIndexRgba	1-82
sceGpIndexRgba#	1-83
sceGpIndexSt#_R	1-85
sceGpIndexSt_R	1-86
sceGpIndexStq#_P	1-87
sceGpIndexStq_P	1-88
sceGpIndexUv	1-89
sceGpIndexUv#	1-90
sceGpIndexXyzf	1-91
sceGpIndexXyzf#	1-92

Set Functions (For Drawing Packets, Some Are Macro Functions)**1-94**

sceGpSetAa1	1-94
sceGpSetAbe	1-95
sceGpSetCtxt	1-96
sceGpSetFog	1-97
sceGpSetRgb	1-98
sceGpSetRgba	1-99
sceGpSetRgbaFM	1-100
sceGpSetStq_P	1-101
sceGpSetStq_R	1-102
sceGpSetUv	1-103
sceGpSetXy	1-104
sceGpSetXyz	1-105
sceGpSetXyz32	1-106
sceGpSetXyzf	1-107
sceGpSetZ32	1-108

Individual Packet Transfer Functions**1-109**

sceGpKickPacket	1-109
sceGpKickPacket2	1-110
sceGpSetPacketMode	1-111
sceGpSyncPacket / sceGpSyncPacketl	1-112

Chain Management Structure

sceGpChain

Chain management structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	January 4, 2002

Structure

```
typedef struct _sceGpChain {
    u_long128 *ot;           Address of ordering table
    u_long128 *pKick;        DMA transfer start address
    u_long128 *pEnd;         DMA transfer end address
    int resolution;          Resolution of ordering table
} sceGpChain;
```

Description

This is the DMA chain management structure.

The addresses kept by pKick and pEnd are the addresses that will be handled by the DMA controller.

Register Unions

sceGpPack

P-format register union

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef union _sceGpPack{
    sceGifPackRgbaq rgbaq;          GIF RGBAQ packing format
    sceGifPackSt st;                GIF ST packing format
    sceGifPackUv uv;                GIF UV packing format
    sceGifPackXyzf xyzf;            GIF XYZF packing format
    sceGifPackXyz xyz;              GIF XYZ packing format
    sceGifPackFog fog;              GIF FOG packing format
    sceGifPackAd ad;                GIF AD packing format
    sceGifPackNop nop;              GIF NOP packing format
    sceVu0FVECTOR fv;              sceVu0FVECTOR-format
    sceVu0IVECTOR iv;              sceVu0IVECTOR-format
    u_long128 ul128;                u_long128 format
    u_long ul[2];                    u_long format
    u_int ui[4];                     u_int format
    float f[4];                      float format
} sceGpPack;
```

Description

This is a register union used by P-format drawing packets.

sceGpReg

R-format register union

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef union _sceGpReg{
    sceGsPrim prim;           PRIM register
    sceGsRgbaq rgbaq;         RGBAQ register
    sceGsSt st;               ST register
    sceGsUv uv;               UV register
    sceGsXyzf xyzf;           XYZF register
    sceGsXyz xyz;             XYZ register
    sceGsTex0 tex0;           TEX0 register
    sceGsClamp clamp;         CLAMP register
    sceGsFog fog;             FOG register
    u_long ul;                u_long format data
    u_int ui[2];              u_int format data
} sceGpReg;
```

Description

This is a register union used by R-format drawing packets.

Packet Structures

sceGpAd

General-purpose register setting packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {  
    sceDmaTag dmanext;           DMA tag for linking the chain  
    sceGifTag giftag;           GIF tag  
    struct {  
        u_long value;           Value of register to be set  
        u_long addr;           GS address of register to be set  
    } reg[1];  
} sceGpAd;
```

Description

This structure represents the structure of a general-purpose register setting packet.

sceGpAlphaEnv

Alpha environment setting packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {
    sceDmaTag dmanext;           DMA tag for linking the chain
    sceGifTag giftag;           GIF tag
    sceGsAlpha alpha;          Value of ALPHA register
    long alphaaddr;            GS address of ALPHA register
    sceGsPabe pabe;           Value of PABE register
    long pabeaddr;            GS address of PABE register
    sceGsTexa texa;          Value of TEXA register
    long texaaddr;            GS address of TEXA register
    sceGsFba fba;           Value of FBA register
    long fbaaddr;            GS address of FBA register
} sceGpAlphaEnv;
```

Description

This structure represents the structure of an alpha environment setting packet.

sceGpCall

call packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {  
    sceDmaTag dmacall;           DMA tag  
    sceDmaTag dmanext;          DMA tag for linking the chain  
} sceGpCall;
```

Description

This structure represents the structure of a call packet.

sceGpLoadImage

Image transfer packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {
    sceDmaTag dmacnt;           DMA tag
    sceGifTag giftag1;          GIF tag
    sceGsBitbltbuf bitbltbuf;   Value of BITBLTBUF register
    long bitbltbufaddr;         GS address of BITBLTBUF register
    sceGsTrxpos trxpos;         Value of TRXPOS register
    long trxposaddr;            GS address of TRXPOS register
    sceGsTrxreg trxreg;         Value of TRXREG register
    long trxregaddr;            GS address of TRXREG register
    sceGsTrxdir trxdir;         Value of TRXDIR register
    long trxdiraddr;            GS address of TRXDIR register
    sceGifTag giftag2;          GIF tag
    sceDmaTag dmaref;           DMA tag
    sceDmaTag dmanext;          DMA tag for linking the chain
    sceGifTag giftag3;          GIF tag
    sceGsTexflush texflush;     Value of TEXFLUSH register
    long texflushaddr;          GS address of TEXFLUSH register
} sceGpLoadImage;
```

Description

This structure represents the structure of an image transfer packet.

sceGpLoadTexelClut

Texture transfer packet structure with CLUT

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {
struct {
    sceDmaTag dmacnt;           DMA tag
    sceGifTag giftag1;         GIF tag
    sceGsBitbltbuf bitbltbuf;
    long bitbltbufaddr;        GS address of BITBLTBUF register
    sceGsTrxpos trxpos;        Value of TRXPOS register
    long trxposaddr;           GS address of TRXPOS register
    sceGsTrxreg trxreg;        Value of TRXREG register
    long trxregaddr;           GS address of TRXREG register
    sceGsTrxdir trxdir;        Value of TRXDIR register
    long trxdiraddr;           GS address of TRXDIR register
    sceGifTag giftag2;         GIF tag
    sceDmaTag dmaref;          DMA tag
} trans[2];
    sceDmaTag dmanext;          DMA tag for linking the chain
    sceGifTag giftag3;          GIF tag
    sceGsTexflush texflush;     Value of TEXFLUSH register
    long texflushaddr;          GS address of TEXFLUSH register
} sceGpLoadTexelClut;
```

Description

This structure represents the structure of a texture transfer packet with a CLUT.

sceGpPrimP

P-format drawing packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {
    sceDmaTag dmanext;           DMA tag for linking the chain
    sceGifTag giftag1;           giftag with PRIM register
    sceGifPackAd userreg;        User-settable register
    sceGifTag giftag2;           giftag
    sceGpPack reg[1];           Drawing parameter register
} sceGpPrimP;
```

Description

This structure represents the structure of a drawing packet of type SCE_GP_PRIM_P. The *dmanext*, *userreg*, *giftag1*, and *giftag2* members are set by the init function.

There can be one or more reg elements. As a result, the size of this structure does not match the actual drawing packet size.

The user must set the *reg* contents correctly by using the index and set functions.

Although a NOP is initially placed in *userreg*, besides using it to set the packet color of a monochrome packet, the user can freely set this register.

Notes

With the initial settings, the contents of *userreg* are treated as a GIF PACKED A+D packet. To change this, modify the REGS0 field of the *giftag1* member appropriately.

sceGpPrimR

R-format drawing packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {
    sceDmaTag dmanext;           DMA tag for linking the chain
    sceGifTag giftag1;           giftag with PRIM register
    sceGifPackAd userreg;        User-settable register
    sceGifTag giftag2;           giftag
    sceGpReg reg[1];             Drawing parameter register
} sceGpPrimR;
```

Description

This structure represents the structure of a drawing packet of type SCE_GP_PRIM_R. The *dmanext*, *userreg*, *giftag1*, and *giftag2* members are set by the init function.

The user must set the *reg* contents correctly by using the index and set functions.

There can be one or more *reg* elements. As a result, the size of this structure does not match the actual drawing packet size.

Although a NOP is initially placed in *userreg*, besides using it to set the packet color of a monochrome packet, the user can freely set this register.

Notes

With the initial settings, the contents of *userreg* are treated as a GIF PACKED A+D packet. To change this, modify the REGS0 field of the *giftag1* member appropriately.

sceGpRef

Ref packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {  
    sceDmaTag dmaref;           DMA tag  
    sceDmaTag dmanext;          DMA tag for linking the chain  
} sceGpRef;
```

Description

This structure represents the structure of a ref packet.

sceGpTexEnv

Texture environment setting packet structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {
    sceDmaTag dmanext;           DMA tag for linking the chain
    sceGifTag giftag;            GIF tag
    sceGsTex1 tex1;              Value of TEX1 register
    long tex1addr;               GS address of TEX1 register
    sceGsTex0 tex0;              Value of TEX0 register
    long tex0addr;               GS address of TEX0 register
    sceGsClamp clamp;            Value of CLAMP register
    long clampaddr;              GS address of CLAMP register
} sceGpTexEnv;
```

Description

This structure represents the structure of a texture environment setting packet.

sceGpTexEnvMipmap

Texture environment setting packet structure with MIPMAP

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct {
    sceDmaTag dmanext;           DMA tag for linking the chain
    sceGifTag giftag;           GIF tag
    sceGsTex1 tex1;             Value of TEX1 register
    long tex1addr;              GS address of TEX1 register
    sceGsTex0 tex0;             Value of TEX0 register
    long tex0addr;              GS address of TEX0 register
    sceGsClamp clamp;           Value of CLAMP register
    long clampaddr;             GS address of CLAMP register
    sceGsMiptbp1 miptbp1;       Value of MIPTBP1 register
    long miptbp1addr;           GS address of MIPTBP1 register
    sceGsMiptbp2 miptbp2;       Value of MIPTBP2 register
    long miptbp2addr;           GS address of MIPTBP2 register
} sceGpTexEnvMipmap;
```

Description

This structure represents the structure of a texture environment setting packet with MIPMAP.

Parameter Structure

sceGpTextureArg

Texture parameter structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	November 5, 2001

Structure

```
typedef struct{
    short tbp;           Texture base pointer (word address/64)
    short tbw;           Texture buffer width (texel unit width/64)
    short tpsm;          Texture pixel storage format
    short tx;            X-offset within texture buffer
    short ty;            Y-offset within texture buffer
    short tw;           Texture width (number of texels)
    short th;           Texture height (number of texels)
    short cbp;          CLUT buffer base pointer (word address/64)
    short cpsm;         CLUT pixel storage format
} sceGpTextureArg;
```

Description

This structure is used as an argument for functions that set texture or image data parameters.

Utility Functions (Some Are Macro Functions)

sceGpChkChainOtSize

Check size of ordering chain (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
#define sceGpChkChainOtSize(  
    r)                                Resolution
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function returns the amount of memory that will be used by the ordering chain for the specified resolution.

Return value

Amount of memory (in quad words) that will be used by the ordering chain for the specified resolution

sceGpChkNumPtoV

Check number of vertices from number of polygons

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

int sceGpChkNumPtoV(

u_int <i>type</i> ,	Packet type
int <i>pnum</i>)	Number of polygons

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function returns the number of vertices corresponding to the number of polygons for the specified packet type.

Return value

Returns the number of vertices.

sceGpChkPacketSize

Check packet size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
Libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpChkPacketSize(
    u_int type,           Packet type
    int arg)              Argument (differs according to type)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function returns the amount of memory to be used by a packet of the specified type and argument. The units are quad words (16 bytes). The value of the *arg* argument is the same as that used for the `sceGpInitPacket()` function. For details, please see the description of that function.

Return value

Returns the amount of memory (in quad words) that the packet will use.

sceGpGetTail

Get packet tail pointer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax**sceDmaTag *sceGpGetTail(****void *pPacket);** Packet starting address**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function gets the packet tail pointer. The tail, where the next packet to be transferred will be connected, will become the position for the DMAnext member in the libgp packet structure.

Return value

Returns the packet tail pointer.

sceGpGetTailChain

Get tail pointer for specified level of chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
sceDmaTag *sceGpGetTailChain(
    sceGpChain *chain,           Pointer to chain management structure
    int level);                 Ordering table level for which pointer is to be obtained
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function gets the tail pointer for the specified level of the chain. The tail, where the next packet to be transferred will be connected, will become the position for each member in the ordering table in the libgp chain management structure.

Return value

Returns the tail pointer for the specified level of the chain.

Chain Manipulation Functions

sceGpAddChain

Register child chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

void sceGpAddChain(

sceGpChain* chain,

Address of parent chain management structure

int level,

(Parent chain) level at which to register child chain

sceGpChain *chain2)

Address of child chain

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function registers a child chain in a parent chain.

Notes

Since the registered child chain is connected to the parent chain, it cannot be transferred independently.

The same child chain cannot be registered more than once simultaneously (regardless of the registration destination).

If you want to call a child chain from two or more locations simultaneously, register it with sceGpCallChain().

Return value

None

sceGpAddChain2

Add child chain after specified position

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
sceDmaTag *sceGpAddChain2(
    sceDmaTag *tail,           Tail pointer of preceding packet
    sceGpChain *chain2);      Pointer to child chain management structure
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function adds a child chain after the specified tail pointer.

Notes

Since the child chain that is added is connected to the parent chain, it cannot be transferred independently.

The same child chain cannot be added more than once at the same time (regardless of where it is being added).

If you want to call a child chain from two or more locations simultaneously, add it using sceGpCallChain().

Return value

Returns the final tail pointer of the added chain.

sceGpAddPacket

Register packet in chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpAddPacket(
    sceGpChain* chain,           Address of chain management structure
    int level,                  Level at which to register packet
    void* p)                    Packet to be added
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function adds a packet to the specified level within the ordering table of a chain.

Notes

The same packet cannot be registered more than once simultaneously (regardless of the registration destination).

If you want to call the same packet multiple times from two or more locations, prepare a child chain, register the relevant packet in it, and use sceGpCallChain() to register multiple calls of the child chain.

Return value

None

sceGpAddPacket2

Add packet after specified position

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
sceDmaTag *sceGpAddPacket2(  
    sceDmaTag *tail,           Tail pointer of preceding packet  
    void* pPacket);           Pointer to packet to be added
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function adds a packet after the specified tail pointer.

Notes

- The same packet cannot be added more than once at the same time (regardless of where it is being added).
- If you want to call the same packet multiple times from two or more locations, prepare a child chain, add the relevant packet to it, then add multiple calls of the child chain using sceGpCallChain() or sceGpCallChain2().

Return value

Returns the tail pointer of the added packet.

sceGpCallChain

Register child chain call

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpCallChain(
    sceGpChain* chain,           Address of parent chain management structure
    int level,                  (Parent chain) level at which to register child chain call
    sceGpChain* chain2,         Address of child chain
    sceGpCall* calltag)         Address of call packet to be used for registration
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function registers a child chain in a parent chain using a CALL invocation.

Notes

Although the *calltag* packet to be used for a call registration is unnecessary for both initialization and configuration, it is required for memory area allocation until the transfer is completed.

Since the registered child chain is connected to the parent chain, it cannot be transferred independently.

A call of the same child chain can be registered any number of times for multiple chains of the same parent chain provided the calls are registered by this function. A child chain registered by this function cannot be registered as a child chain of the `sceGpAddChain()` function.

The call depth due to CALL tags of chain call registrations and call packets is limited to at most two levels.

Return value

None

sceGpCallChain2

Add child chain call after specified position

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
sceDmaTag *sceGpCallChain2(  
    sceDmaTag *tail,           Tail pointer of preceding packet  
    sceGpChain* chain2,       Pointer to chain structure where child chain call is to be added  
    sceGpCall* callp);        Pointer to call packet to be used for the add
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function adds a child chain after the specified tail pointer through a call invocation. One CALL packet is required for the add.

Notes

- Although the CALL packet used for adding the call isn't necessary for either initialization or configuration, it is needed for memory allocation up until the transfer completes.
- Since the child chain that is added is connected to the parent chain, it cannot be transferred independently.
- This function differs from the sceAddChain() and sceAddChain2() functions in that a call of the same child chain can be added from more than one parent chain.
- A child chain added by this function cannot be added again as a child chain using the sceGpAddChain() or sceGpAddChain2() functions.
- The call depth from CALL tags of chain call adds and call packets is limited to at most two levels.

Return value

Returns the tail pointer of the call packet that was used for the add.

sceGpEndLevel

Set chain transfer end level

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
void sceGpSetEndLevel(
    sceGpChain *chain,           Address of chain management structure
    int level)                  Level to be interrupted
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets up the chain so that it ends at an intermediate level of the ordering table.

Packets or chains that were added at or after the level indicated by the *level* argument are not transferred.

The contents that had been set in the original interrupted level (such as the DMA tag for the next packet) are copied to the newly set level. Also, the DMA tag of the original interrupted level is returned in the NEXT tag.

Return value

None

sceGpInitChain

Initialize chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitChain(
    sceGpChain *chain,           Address of chain management structure
    void *addr,                 Address of ordering table
    int resolution)             Resolution of ordering table
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function initializes the chain management structure, then calls the sceGpResetChain() function.

Notes

The return value is the amount of memory to be used by the ordering table. This is not the memory size of the chain management structure itself. The memory areas required for the chain management structure and the ordering table must be allocated externally with appropriate alignment.

Return value

Returns the amount of memory (in quad words) to be used by the ordering table.

sceGpKickChain

Transfer chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpKickChain(
    sceGpChain *chain,           Address of chain management structure
    int path)                   Path used
                                SCE_GP_PATH1: PATH1
                                SCE_GP_PATH2: PATH2
                                SCE_GP_PATH3: PATH3
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function checks for an available DMA and starts the DMA transfer of a chain. It returns without waiting for the end of the transfer.

The registered packet or contents of the chain cannot be changed and the chain cannot be reset until the transfer ends.

Notes

When PATH1 or PATH2 is specified, DMA channel 1 is used with TTE=1 (Tag Transfer Enable ON).

When PATH3 is specified, DMA channel 2 is used with TTE=0 (Tag Transfer Enable OFF).

When PATH1 and PATH 2 packets are mixed, either path can be specified.

The end of the transfer can be detected by using the sceGsSyncPath() function, for example.

The D cache should be flushed, etc. before calling this function, as required.

Return value

Table 1-1

Value	Result
0	When transfer started
-1	When processing for starting the transfer failed because the DMA channel to be used for the transfer was busy

sceGpKickChain2

Transfer chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
int sceGpKickChain2(  
    sceGpChain *chain,           Address of chain management structure  
    int path)                   Path used  
                                SCE_GP_PATH1: PATH1  
                                SCE_GP_PATH2: PATH2  
                                SCE_GP_PATH3: PATH3
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function starts the DMA transfer of a chain without confirming that the DMA channel is available. It returns without waiting for the end of transfer.

This function should be called after confirming beforehand that a DMA channel is available.

The contents of the added packet or chain cannot be changed and the chain cannot be reset until the transfer ends.

Notes

- When PATH1 or PATH2 is specified, DMA channel 1 is used with TTE=1 (Tag Transfer Enable ON).
- When PATH3 is specified, DMA channel 2 is used with TTE=0 (Tag Transfer Enable OFF).
- When PATH1 and PATH 2 packets are mixed, either path can be specified.
- The end of transfer can be detected by using the sceGsSyncPath() function, for example.
- This function should be called after flushing the D cache, if necessary.

Return value

None

sceGpPrintChain

Output chain contents to console

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpPrintChain(
    sceGpChain *chain,           Address of chain management structure
    int verbosity,              Display detail
                                0: Only the amount for each packet
                                1: Also display packet types sequentially
    int from,                  Display starting packet number (use the numbers
                                displayed for verbosity=1)
    int num)                   Number of packets to be displayed (0: Until the last
                                packet)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This is a debugging function for displaying chains and registered packets or chains.

Invalid results may be obtained when call packets are used or when non-libgp packets or chains are added.

The display of the number of DMA tags is the result of analyzing the entire chain, regardless of the arguments.

Return value

None

sceGpRemoveNextChain

Remove chain from parent chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

sceGpChain* sceGpRemoveNextChain(
sceDmaTag *tail,
sceGpChain *chain)

Pointer to tail of preceding packet

Pointer to management structure of chain to be removed

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function removes a previously added child chain from a chain. For the *tail* argument, specify a pointer to the “tail” of the preceding packet (DMAnext member of the packet structure). The tail can be obtained by using the sceGpSearchTailToRemove() function. If the preceding packet is known, the tail can also be obtained by using a function such as sceGpGetTailChain() or sceGpGetTail().

This function is only valid for child chains that were added with sceGpAddChain() or sceGpAddChain2().

This function cannot be used for a child chain that was added with sceGpCallChain(). In that case, delete the Call packet that was used when the child chain was added. If this is done, the child chain will no longer be able to be called.

Return value

Returns a pointer to the chain that was removed.

sceGpRemoveNextPacket

Remove packet from chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
sceGpChain* sceGpRemoveNextPacket(
    sceDmaTag *tail,)           Pointer to tail of preceding packet
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function removes a previously registered packet from a chain. For the *tail* argument, specify a pointer to the “tail” of the preceding packet (DMAnext member of the packet structure). The tail can be obtained by using the sceGpSearchTailToRemove() function. If the preceding packet is known, the tail can also be obtained by using a function such as sceGpGetTailChain() or sceGpGetTail().

Return value

Returns a pointer to the packet that was removed.

sceGpResetChain

Reset chain ordering table

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpResetChain(
    sceGpChain *chain)           Address of chain management structure
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function initializes the ordering table and sets the pKick and pEnd members of the chain.

Notes

The packet or separate chain registrations that existed prior to the reset are all invalidated. The chain start level (sceGpSetStartLevel) and interruption process (sceGpTermChain()) are also invalidated.

Return value

None

sceGpResetChainRev

Reverse reset chain ordering table

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpResetChainRev(
    sceGpChain *chain)           Address of chain management structure
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function initializes the ordering table in the reverse direction (from the end to the beginning).

Notes

When a reset is performed by this function, packets are transferred beginning with those registered at a higher level.

Within the same level, packets are transferred beginning with those that were registered last, in a similar manner as with a forward reset.

Return value

None

sceGpSearchTailToRemove

Get tail pointer for removal

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

sceDmaTag* sceGpSearchTailToRemove(

sceGpChain *chain, Address of chain management structure

void* packet) Pointer to packet to be removed

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function gets a pointer to the tail of the preceding packet, which is to be used by sceGpRemoveNextPacket() or sceGpRemoveNextChain() when removing a packet from the chain.

When the object to be removed is a chain, the pKick member of the child chain should be specified for the *packet* argument. For example, if pChain is a pointer to the management structure of a child chain, pChain->pKick should be specified for the *packet* argument. However, for a child chain call that was added using sceGpCallChain() or sceGpCallChain2(), a pointer to the Call packet that was used during the add should be specified.

Since this function searches for the required tail pointer by scanning from the beginning of the chain, it can take quite a long time when the chain is particularly long. As a result, either try not to perform a remove in this case, or if it is absolutely necessary to perform a remove, do not use this function. It will be more efficient to remember beforehand the required tail pointer or preceding packet during the add.

Return value

Returns a tail pointer to be used when removing the specified packet.

sceGpSetStartLevel

Set chain start level

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**void sceGpSetStartLevel(**

sceGpChain * <i>chain</i> ,	Address of chain management structure
int <i>level</i>)	Level where intermediate start is to be set

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets that a chain is to start at an intermediate level of the ordering table.

Packets or chains that were registered at a level before the level indicated by the *level* argument are not transferred. (Packets that were registered at the specified level are transferred.)**Return value**

None

sceGpTermChain

Set chain interruption

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpTermChain(
    sceGpChain *chain,           Address of chain management structure
    int level,                   Level where chain is to be interrupted
    int isret)                   0: Interrupt chain due to END tag
                                1: Interrupt chain due to RET tag (for Call invocation)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This is an old function that has been retained for compatibility. Normally, the sceGpSetEndLevel() function should be used.

This function sets that a chain is to end at an intermediate level of the ordering table.

Packets or chains that were registered after the level specified by the *level* argument are not transferred. (Packets that were registered at the specified level are also not transferred.)

A level higher than the original level cannot be set. Furthermore, this function cannot be used to manipulate a child chain that was added with sceAddChain() or sceAddChain2().

Return value

None

Packet Control Functions

sceGpCopyPacket

Copy packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpCopyPacket(
    void* dp,           Destination packet
    void* sp)           Source packet
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function copies a packet.

The init function must already have been called for the source packet.

The init or set function need not be called for the destination packet after copying.

Notes

The memory area required for the destination packet must be allocated in advance externally with appropriate alignment.

Return value

Returns the copied memory size (in quad words).

sceGpInsertPacket

Add packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

void sceGpInsertPacket(

void* *pa*, Packet located immediately before the add point

void* *pb*) Packet to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function adds a new packet (*pb*) immediately after the specified packet (*pa*).

Notes

This function is used to insert a packet at an intermediate location within a chain.

Return value

None

Packet Default Value Setting Functions

sceGpSetDefaultAa1

Set AA1 default value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetDefaultAa1(
```

```
int aa1)
```

Default AA1 bit (0: AA1 ON, 1: AA1 OFF)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the AA1 (one-pass antialiasing) value to be used by a drawing packet in a subsequent init function. The default value is 0 (OFF).

Notes

The value set here is valid only for an init function that is called after this value is set. To change the AA1 bit of a drawing packet for which the init function was already loaded, use the `sceGpSetAa1()` macro function.

Return value

None

sceGpSetDefaultAbe

Set ABE default value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**void sceGpSetDefaultAbe(****int *abe*)**

Default ABE bit

0: Alpha blending OFF

1: Alpha blending ON

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the ABE (alpha blending enable) value to be used by a drawing packet in a subsequent init function. The default value is 0 (OFF).

Notes

The value set here is valid only for an init function that is called after this value is set. To change the ABE bit of a drawing packet for which the init function was already loaded, use the `sceGpSetAbe()` macro function.

Return value

None

sceGpSetDefaultCtxt

Set default value of context used

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetDefaultCtxt(
```

int <i>ctxt</i>)	Default context (0: CTXT1, 1: CTXT2)
--------------------------	--------------------------------------

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the context to be used by a drawing packet in a subsequent init function. If 0 is set, CTXT1 will be used, and if 1 is set, CTXT2 will be used. The default value is 0 (CTXT1).

Notes

The value set here only affects the PRIM register of the drawing packet. The context of a non-drawing packet such as a texture environment setting packet or alpha environment setting packet must be set by another method.

The value set here is valid only for an init function that is called after this value is set. To change the context to be used of a drawing packet for which the init function was already loaded, use the `sceGpSetCtxt()` macro function.

Return value

None

sceGpSetDefaultDirectHL

Set default level of direct command

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**void sceGpSetDefaultDirectHL(****int on)**

0: Use Direct command, 1: Use DirectHL command

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function controls the command to be used by packets when transferring PATH2 data in a subsequent init function. The default value is 0 (use Direct command).

Notes

For information about the differences between the Direct and DirectHL commands, refer to the VPU chapter of the EE User's Manual.

Return value

None

sceGpSetDefaultFog

Set FGE default value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetDefaultFog(
```

int fge)

Default FGE bit (0: Fogging OFF, 1: Fogging ON)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the FGE (fogging enable) value to be used by a drawing packet in a subsequent init function. The default value is 0 (OFF).

Notes

The value set here is valid only for an init function that is called after this value is set. To change the FGE bit of a drawing packet for which the init function was already loaded, use the `sceGpSetFge()` macro function.

Return value

None

sceGpSetDefaultZ32

Set default register used by drawing packets

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.5	May 10, 2002

Syntax

void sceGpSetDefaultZ32(

int on) Specify 0 to use XYZF2 in drawing packets, and 1 to use XYZ2

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function provides a mechanism that enables future init functions to use either XYZF2 or XYZ2 in drawing packets. The initial value is 0 (use XYZF2).

Notes

The setting here is effective only for init functions which are subsequently called. Use sceGpSetZ32() to change the register descriptor of a drawing packet that has already read an init function.

Return value

None

Packet Initialization Functions

sceGpInitAd

Initialize general-purpose register setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitAd(
    sceGpAd *p,           Address for creating packet
    int num)              Number of register settings
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes a general-purpose register setting packet at the specified address. The number of registers to be set is given as an argument.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

The size of the general-purpose register setting packet differs from the size of the sceGpAd structure.

Among the members of the sceGpAd structure that are set by this function are giftag and part of dmanext.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitAlphaEnv

Initialize alpha environment setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitAlphaEnv(
    sceGpAlphaEnv *p,           Address for creating packet
    int ctxt)                   Context to be set (0: CTXT1, 1: CTXT2)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes an alpha environment setting packet at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Among the members of the sceGpAlphaEnv structure that are set by this function are alphaaddr, pabeaddr, texaaddr, fbaaddr and part of dmanext.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitCall

Initialize call packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitCall(
    sceGpCall *p)           Address for creating packet
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes a call packet at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Among the members of the sceGpCall structure that are set by this function are part of dmacall and part of dmanext.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitLoadImage

Initialize image transfer packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitLoadImage(
    sceGpLoadImage *p)           Address for creating packet
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes an image transfer packet at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Among the members of the sceGpInitLoadImage structure that are set by this function are dmacnt, giftag1, bitbltbufaddr, txposaddr, txregaddr, txdiraddr, giftag2, part of dmaref, part of dmanext, giftag3, texflush, and texflushaddr.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitLoadTexelClut

Initialize texture transfer packet with CLUT

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitLoadTexelClut(
    sceGpLoadTexelClut *p)           Address for creating packet
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes a texture transfer packet with CLUT at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Among the members of the sceGpLoadTexelClut structure that are set by this function are trans[0].dmacnt, trans[0].giftag1, trans[0].giftag2, part of trans[0].dmaref, trans[0].bitbltbufaddr, trans[0].trxposaddr, trans[0].trxregaddr, trans[0].trxdiraddr, trans[1].dmacnt, trans[1].giftag1, trans[1].giftag2, part of trans[1].dmaref, trans[1].bitbltbufaddr, trans[1].trxposaddr, trans[1].trxregaddr, trans[1].trxdiraddr, part of dmanext, giftag3, texflush, and texflushaddr.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitPacket

Initialize packet (general)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitPacket(
    void *p,           Address for creating packet
    u_int type,        Packet type
    int arg)           Argument
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe(does not depend on interrupt enabled or disabled state)

Description

This function calls one of the following functions based on the packet type to initialize a packet. For details about the arguments and operations, refer to the corresponding function reference.

Table 1-2

Type	Called Function
SCE_GP_PRIM_RI subtype	sceGpInitPrimR((sceGpPrimR*)p, type, arg)
SCE_GP_PRIM_PI subtype	sceGpInitPrimP((sceGpPrimP*)p, type, arg)
SCE_GP_ALPHAENV	sceGpInitAlphaEnv((sceGpAlphaEnv*)p, arg);
SCE_GP_TEXENV	sceGpInitTexEnv((sceGpTexEnv*)p, arg);
SCE_GP_TEXENVMIPMAP	sceGpInitTexEnvMipmap ((sceGpTexEnvMipmap*)p, arg);
SCE_GP_LOADIMAGE	sceGpInitLoadImage((sceGpLoadImage*)p);
SCE_GP_LOADTEXELCLUT	sceGpInitLoadTexelClut ((sceGpLoadTexelClut*)p);
SCE_GP_AD	sceGpInitAd((sceGpAd*)p, arg);
SCE_GP_REF	sceGpInitRef((sceGpRef*)p);
SCE_GP_CALL	sceGpInitCall((sceGpCall*)p);

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitPrimP

Initialize P-format drawing packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitPrimP(
    sceGpPrimP *p,           Address for creating packet
    u_int type,              Packet type
    int pnum)                Number of polygons
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe(does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes a P-format drawing packet at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

The size of a drawing packet differs from the size of the sceGpPrimP structure.

Among the members of the sceGpPrimP structure that are set by this function are giftag1, userreg, giftag2, and part of dmanext. userreg is set to NOP.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitPrimR

Initialize R-format drawing packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitPrimR(
    sceGpPrimR *p,           Address for creating packet
    u_int type,              Packet type
    int pnum)                Number of polygons
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe(does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes an R-format drawing packet at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

The size of a drawing packet differs from the size of the sceGpPrimR structure.

Among the members of the sceGpPrimR structure that are set by this function are giftag1, userreg, giftag2, and part of dmanext. userreg is set to NOP.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitRef

Initialize ref packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitRef(
    sceGpRef *p)           Address for creating packet
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes a ref packet at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Among the members of the sceGpRef structure that are set by this function are part of dmaref and part of dmanext.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitTexEnv

Initialize texture environment setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
int sceGpInitTexEnv(
    sceGpTexEnv *p,           Address for creating packet
    int ctxt)                 Context to be set (0: CTXT1, 1: CTXT2)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes a texture environment setting packet at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Among the members of the sceGpTexEnv structure that are set by this function are giftag, tex1addr, tex0addr, clampaddr, and part of dmanext.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

sceGpInitTexEnvMipmap

Initialize texture environment setting packet with MipMap

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

int sceGpInitTexEnvMipmap(

sceGpTexEnvMipmap *p, Address for creating packet
 int ctxt) Context to be set (0: CTXT1, 1: CTXT2)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function creates and initializes a texture environment setting packet with MipMap at the specified address.

Notes

The memory area required for the packet must be allocated in advance externally with appropriate alignment.

Among the members of the sceGpTexEnvMipmap structure that are set by this function are giftag, tex1addr, tex0addr, clampaddr, miptbpaddr, miptbp2addr, and part of dmanext.

Return value

Returns the amount of memory (in quad words) to be used by the packet.

Set Functions (For Non-Drawing Packets)

sceGpSetAd

Set general-purpose register setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetAd(
    sceGpAd *p,           Address of packet to be set
    int index,            Register number
    u_long addr,          Address of GS register to be set
    u_long value)         Value to be set
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the register to be set at the specified *index* position within the packet.

Return value

None

sceGpSetAlphaEnv

Set alpha environment setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetAlphaEnv(
    sceGpAlphaEnv* p,           Address of packet to be set
    int func,                   Alpha blending function (described later)
    int fix)                    FIX value
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets a packet so that the specified function will be executed.

Table 1-3

Setting	Output Color	Blending Function
SCE_GP_ALPHA_NOP	Source color as is	(Cs-0)*128+0
SCE_GP_ALPHA_INTER_AS	Color obtained by interpolating source color and destination color using As	(Cs-Cd)@As+Cd
SCE_GP_ALPHA_INTER_AD	Color obtained by interpolating source color and destination color using Ad	(Cs-Cd)@Ad+Cd
SCE_GP_ALPHA_INTER_FIX	Color obtained by interpolating source color and destination color using FIX	(Cs-Cd)@FIX+Cd
SCE_GP_ALPHA_RINTER_AS	Color obtained by interpolating source color and destination color using (128-As)	(Cd-Cs)@As+Cs
SCE_GP_ALPHA_RINTER_AD	Color obtained by interpolating source color and destination color using (128-Ad)	(Cd-Cs)@Ad+Cs
SCE_GP_ALPHA_RINTER_FIX	Color obtained by interpolating source color and destination color using (128-FIX)	(Cd-Cs)@FIX+Cs
SCE_GP_ALPHA_ADD	Color obtained by adding source color and destination color	(Cs-0)@128+Cd
SCE_GP_ALPHA_ADD_CS_FIX	Color obtained by multiplying source color by FIX and adding it (to destination color)	(Cs-0)@FIX+Cd
SCE_GP_ALPHA_ADD_CD_FIX	Color obtained by multiplying destination color by FIX and adding it (to source color)	(Cd-0)@128+Cs
SCE_GP_ALPHA_ADD_CS_AS	Color obtained by multiplying source color by As and adding it (to destination color)	(Cs-0)@As+Cd

Setting	Output Color	Blending Function
SCE_GP_ALPHA_ADD_CD_AS	Color obtained by multiplying destination color by As and adding it (to source color)	(Cd-0)@As+Cs
SCE_GP_ALPHA_ADD_CS_AD	Color obtained by multiplying source color by Ad and adding it (to destination color)	(Cs-0)@Ad+Cd
SCE_GP_ALPHA_ADD_CD_AD	Color obtained by multiplying destination color by Ad and adding it (to source color)	(Cd-0)@Ad+Cs
SCE_GP_ALPHA_SUB_CS	Color obtained by subtracting source color (from destination color)	(0-Cs)@128+Cd
SCE_GP_ALPHA_SUB_CD	Color obtained by subtracting destination color (from source color)	(0-Cd)@128+Cs
SCE_GP_ALPHA_SUB_CS_FIX	Color obtained by multiplying source color by FIX and subtracting it (from destination color)	(0-Cs)@FIX+Cd
SCE_GP_ALPHA_SUB_CD_FIX	Color obtained by multiplying destination color by FIX and subtracting it (from source color)	(0-Cd)@FIX+Cs
SCE_GP_ALPHA_SUB_CS_AS	Color obtained by multiplying source color by As and subtracting it (from destination color)	(0-Cs)@As+Cd
SCE_GP_ALPHA_SUB_CD_AS	Color obtained by multiplying destination color by As and subtracting it (from source color)	(0-Cd)@As+Cs
SCE_GP_ALPHA_SUB_CS_AD	Color obtained by multiplying source color by Ad and subtracting it (from destination color)	(0-Cs)@Ad+Cd
SCE_GP_ALPHA_SUB_CD_AD	Color obtained by multiplying destination color by Ad and subtracting it (from source color)	(0-Cd)@Ad+Cs
SCE_GP_ALPHA_MUL_CS_AS	Color obtained by multiplying source color by As	(Cs-0)@As+0
SCE_GP_ALPHA_MUL_CS_AD	Color obtained by multiplying source color by Ad	(Cs-0)@Ad+0
SCE_GP_ALPHA_MUL_CS_FIX	Color obtained by multiplying source color by FIX	(Cs-0)@FIX+0
SCE_GP_ALPHA_MUL_CD_AS	Color obtained by multiplying destination color by As	(Cd-0)@As+0
SCE_GP_ALPHA_MUL_CD_AD	Color obtained by multiplying destination color by Ad	(Cd-0)@Ad+0
SCE_GP_ALPHA_MUL_CD_FIX	Color obtained by multiplying destination color by FIX	(Cd-0)@FIX+0

Notes

In the alpha blending multiplication, 128 corresponds to 1.0. In addition, the operation described as “multiplying X by Y” and denoted as “X @ Y” represents the operation $(X * Y) >> 7$.

The functions that can be set using the variables given here do not include all of the alpha blending types that are available in the GS.

To use a setting other than the ones given here, overwrite the packet contents after the value has been set.

The PABE and FBA registers will be set to 0. To change these values, overwrite the packet contents after these values are set.

Since context switching cannot be performed for the PABE and FBE registers, watch out for contention if you do use context switching.

Return value

None

sceGpSetAlphaEnvFunc

Set alpha blending function of alpha environment setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetAlphaEnvFunc(
    sceGpAlphaEnv* p,           Address of packet to be set
    int func,                   Alpha blending function (described later)
    int fix)                    FIX value
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets a packet so that the specified function will be executed.

Table 1-4

Setting Value	Output Color
SCE_GP_ALPHA_NOP	Source color as is
SCE_GP_ALPHA_INTER_AS	Color obtained by interpolating source color and destination color using As
SCE_GP_ALPHA_INTER_AD	Color obtained by interpolating source color and destination color using Ad
SCE_GP_ALPHA_INTER_FIX	Color obtained by interpolating source color and destination color using FIX
SCE_GP_ALPHA_RINTER_AS	Color obtained by interpolating source color and destination color using (128-As)
SCE_GP_ALPHA_RINTER_AD	Color obtained by interpolating source color and destination color using (128-Ad)
SCE_GP_ALPHA_RINTER_FIX	Color obtained by interpolating source color and destination color using (128-FIX)
SCE_GP_ALPHA_ADD	Color obtained by adding source color and destination color
SCE_GP_ALPHA_ADD_CS_FIX	Color obtained by multiplying source color by FIX and adding it (to destination color)
SCE_GP_ALPHA_ADD_CD_FIX	Color obtained by multiplying destination color by FIX and adding it (to source color)
SCE_GP_ALPHA_ADD_CS_AS	Color obtained by multiplying source color by As and adding it (to destination color)
SCE_GP_ALPHA_ADD_CD_AS	Color obtained by multiplying destination color by As and adding it (to source color)
SCE_GP_ALPHA_ADD_CS_AD	Color obtained by multiplying source color by Ad and adding it (to destination color)

Setting Value	Output Color
SCE_GP_ALPHA_ADD_CD_AD	Color obtained by multiplying destination color by Ad and adding it (to source color)
SCE_GP_ALPHA_SUB_CS	Color obtained by subtracting source color (from destination color)
SCE_GP_ALPHA_SUB_CD	Color obtained by subtracting destination color (from source color)
SCE_GP_ALPHA_SUB_CS_FIX	Color obtained by multiplying source color by FIX and subtracting it (from destination color)
SCE_GP_ALPHA_SUB_CD_FIX	Color obtained by multiplying destination color by FIX and subtracting it (from source color)
SCE_GP_ALPHA_SUB_CS_AS	Color obtained by multiplying source color by As and subtracting it (from destination color)
SCE_GP_ALPHA_SUB_CD_AS	Color obtained by multiplying destination color by As and subtracting it (from source color)
SCE_GP_ALPHA_SUB_CS_AD	Color obtained by multiplying source color by Ad and subtracting it (from destination color)
SCE_GP_ALPHA_SUB_CD_AD	Color obtained by multiplying destination color by Ad and subtracting it (from source color)
SCE_GP_ALPHA_MUL_CS_AS	Color obtained by multiplying source color by As
SCE_GP_ALPHA_MUL_CS_AD	Color obtained by multiplying source color by Ad
SCE_GP_ALPHA_MUL_CS_FIX	Color obtained by multiplying source color by FIX
SCE_GP_ALPHA_MUL_CD_AS	Color obtained by multiplying destination color by As
SCE_GP_ALPHA_MUL_CD_AD	Color obtained by multiplying destination color by Ad
SCE_GP_ALPHA_MUL_CD_FIX	Color obtained by multiplying destination color by FIX

Notes

In the calculation of alpha blending, 128 corresponds to 1.0, and "multiplying X by Y" indicates the operation $(X * Y) \gg 7$.

The functions that can be set by using the variables given here do not include all types of alpha blending that are possible in the GS.

To use a setting other than the ones given here, overwrite the contents of a separate packet.

The setting values of the PABE, TEXA and FBA registers are not changed.

Return value

None

sceGpSetCall

Set call packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**void sceGpSetCall(****sceGpCall *p,**

Address of packet to be set

void* addr)

Address of DMA chain to be called using call

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the contents of a call packet. The ret tag must be appended to the end of the DMA chain that is called when the chain is transferred.

Return value

None

sceGpSetLoadImage

Set image transfer packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetLoadImage(
    sceGpLoadImage *p,           Address of packet to be set
    sceGpTextureArg *texarg,     Texture parameter structure
    void *srcaddr,               Address of image data in main memory
    int isClut)                  0: Texel data,
                                1: CLUT data
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses the texture parameter structure settings to set the contents of a packet.

Notes

CLUT data is not rearranged. It is transferred to the GS in its original order.

The image data address must be 128-bit aligned.

Return value

None

sceGpSetLoadImageByArgTim2

Use both texture parameter structure and TIM2 to set image transfer packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetLoadImageByArgTim2(
    sceGpLoadImage *p,           Address of packet to be set
    const sceGpTextureArg *arg,   Texture parameter structure
    const void *ptim2,           Address of TIM2 data
    int picture,                 Picture number to be used
    int miplevel,                Mipmap level to be used
    int isClut)                  0: Texel data
                                1: CLUT data
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses the texture parameter structure and the settings within TIM2 data to set the contents of a packet. If a setting exists in both the texture parameter structure and TIM2 data, the texture parameter structure value takes precedence. However, the TIM2 data setting is used when the value of the texture parameter structure member is negative.

Notes

CLUT data is not rearranged. It is transferred to the GS in its original order.

Return value

None

sceGpSetLoadImageByTim2

Use TIM2 to set image transfer packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetLoadImageByTim2(
    sceGpLoadImage *p,           Address of packet to be set
    const void *ptim2,           Address of TIM2 data
    int picture,                  Picture number to be used
    int miplevel,                 Mipmap level to be used
    int isClut)                   0: Texel data
                                1: CLUT data
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses settings within TIM2 data to set the contents of a packet.

Notes

CLUT data is not rearranged. It is transferred to the GS in its original order.

Return value

None

sceGpSetLoadTexelClut

Set texture transfer packet with CLUT

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetLoadTexelClut(
    sceGpLoadTexelClut *p,           Address of packet to be set
    sceGpTextureArg *texarg,         Texture parameter structure
    void *srcaddr,                   Address of texel data in main memory
    void *csrcaddr)                  Address of CLUT data in main memory
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses texture parameter structure settings to set the contents of a packet.

Notes

CLUT data is not rearranged. It is transferred to the GS in its original order.

The texel and CLUT data addresses must be 128-bit aligned.

Return value

None

sceGpSetLoadTexelClutByArgTim2

Use both texture parameter structure and TIM2 to set texture transfer packet with CLUT

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	July 8, 2002

Syntax

```
void sceGpSetLoadTexelClutByArgTim2(
    sceGpLoadTexelClut *p,           Address of packet to be set
    const sceGpTextureArg *arg,      Texture parameter structure
    const void *ptim2,               Address of TIM2 data
    int picture,                     Picture number to be used
    int miplevel                     Mipmap level to be used
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses the texture parameter structure and the settings within TIM2 data to set the contents of a packet. If a setting exists in both the texture parameter structure and TIM2 data, the texture parameter structure value takes precedence. However, the TIM2 data setting is used when the value of the texture parameter structure member is negative.

Notes

CLUT data is not rearranged. It is transferred to the GS in its original order.

Return value

None

sceGpSetLoadTexelClutByTim2

Use TIM2 to set texture transfer packet with CLUT

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetLoadTexelClutByTim2(
    sceGpLoadTexelClut *p,           Address of packet to be set
    const void *ptim2,               Address of TIM2 data
    int picture,                      Picture number to be used
    int miplevel)                    Mipmap level to be used
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses settings within TIM2 data to set the contents of a packet.

Notes

CLUT data is not rearranged. It is transferred to the GS in its original order.

Return value

None

sceGpSetRef

Set ref packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**void sceGpSetRef(**

sceGpRef *p,	Address of packet to be set
void* addr,	Transfer data address
int size,	Transfer size (in quad words)
int path)	Path used
	SCE_GP_PATH1: PATH1
	SCE_GP_PATH2: PATH2
	SCE_GP_PATH3: PATH3

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the contents of a ref packet.

When the transfer data uses PATH1 (when a VIF command or data to be used in VUMEM or VU1 is entered), PATH1 should be set for the path argument.

If PATH2 or PATH3 is specified, a Direct or DirectHL command is inserted in the highest 32 bits of the dmaref member to pass data from VIF1 to the GIF.

Note that with libgp, if PATH1 or PATH2 is used when transferring a chain, the TTE (Tag Transfer Enable) bit will be ON.

Return value

None

sceGpSetTexEnv

Set texture environment setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetTexEnv(
    sceGpTexEnv *p,           Address of packet to be set
    sceGpTextureArg *texarg,  Texture parameter structure
    int tfx,                  Texture function
                              0: MODULATE
                              1: DECAL
                              2: HIGHLIGHT
                              3: HIGHLIGHT2
    int filter)               Texture filter
                              0: Point sampling NEAREST
                              1: Bilinear sampling LINEAR
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses the texture parameter structure settings to set the packet contents.

Notes

The wrap mode for the clamp register is set to CLAMP or REGION_CLAMP according to the texture size.

When REGION_CLAMP is used, the upper and lower limit clamp values are also set according to the texture size.

To use REPEAT mode, overwrite the value in the appropriate packet.

The maximum MIP level is set to 0.

The texture color component TCC value is set to 1. To use 0, overwrite the value in the appropriate packet.

Return value

None

sceGpSetTexEnvByArgTim2

Use both texture parameter structure and TIM2 to set texture environment setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetTexEnvByArgTim2(
    sceGpTexEnv *p,           Address of packet to be set
    const sceGpTextureArg *texarg, Texture parameter structure
    const void *ptim2,        Address of TIM2 data
    int picture,              Picture number to be used
    int miplevel)             Mipmap level to be used
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses the texture parameter structure and the settings within TIM2 data to set the contents of a packet. If a setting exists in both the texture parameter structure and TIM2 data, the texture parameter structure value takes precedence. However, the TIM2 data setting is used when the value of the texture parameter structure member is negative.

Notes

The wrap mode for the clamp register is set to CLAMP or REGION_CLAMP according to the texture size.

When REGION_CLAMP is used, the upper and lower limit clamp values are also set according to the texture size.

To use REPEAT mode, overwrite the value in the appropriate packet.

The maximum MIP level is set to 0.

Return value

None

sceGpSetTexEnvByDrawEnv

Set up texture environment configuration packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax**void sceGpSetTexEnvByDrawEnv(****sceGpTexEnv** *p, Address of packet to be set up**sceGsDrawEnv1** *draw, Drawing environment**int** tcc, Texture color component**int** tfx, Texture function

0: MODULATE

1: DECAL

2: HIGHLIGHT

3: HIGHLIGHT2

int filter); Texture filter

0: Point sampling NEAREST

1: Bilinear sampling LINEAR

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets up the contents of the packet so that the specified drawing environment will be used as the texture.

Notes

The wrap mode for the clamp register is set to CLAMP or REGION_CLAMP from the value of the scissor register and buffer width of the original drawing environment.

When REGION_CLAMP mode is used, the upper and lower limit clamp values are also set according to the texture size.

To use REPEAT mode, overwrite the value in the appropriate packet.

TW and TH are set with the minimum values that include the region indicated by the scissor register, for the drawing environment.

The maximum MIP level is set to 0.

The value of TCC (texture color component) is set to 1. To use 0, overwrite the value in the appropriate packet.

Return value

None

sceGpSetTexEnvByTim2

Use TIM2 to set texture environment setting packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetTexEnvByTim2(
    sceGpTexEnv *p,           Address of packet to be set
    const void *ptim2,        Address of TIM2 data
    int picture,              Picture number to be used
    int miplevel)             Mipmap level to be used
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function uses settings within TIM2 data to set the contents of a packet.

Notes

The wrap mode for the clamp register is set to CLAMP or REGION_CLAMP according to the texture size.

When REGION_CLAMP is used, the upper and lower limit clamp values are also set according to the texture size.

To use REPEAT mode, overwrite the value in the appropriate packet.

The maximum MIP level is set to 0.

Return value

None

Index Acquisition Functions (For Drawing Packets)

sceGpIndexQ#_R

Get Q index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
unsigned int sceGpIndexQ#_R(
    unsigned int n)           Relative position of Q value
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function returns the index for which the n -th Q value should be set for an R-format drawing packet.

Notes

A string corresponding to the subtype is entered for #.

This section includes the following functions.

sceGpIndexQLineFTS_R(), sceGpIndexQLineGTS_R()
 sceGpIndexQLineStripFTS_R(), sceGpIndexQLineStripGTS_R()
 sceGpIndexQPointFTS_R(), sceGpIndexQSpriteFTS_R()
 sceGpIndexQTriFTS_R(), sceGpIndexQTriFanFTS_R()
 sceGpIndexQTriFanGTS_R(), sceGpIndexQTriGTS_R()
 sceGpIndexQTriStripFTS_R(), sceGpIndexQTriStripGTS_R()

Return value

Returns the index for which n -th Q value should be set.

sceGpIndexQ_R

Get Q index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**unsigned int sceGpIndexQ_R(**

unsigned int *type*, Packet type
unsigned int *n*) Relative position of Q value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the *n*-th Q value should be set for an R-format drawing packet.**Return value**Returns the index for which *n*-th Q value should be set.

sceGpIndexRgba

Get RGBA index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**unsigned int sceGpIndexRgba(**

unsigned int <i>type</i> ,	Packet type
unsigned int <i>n</i>)	Relative position of RGBA value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the *n*-th RGBA value should be set.**Return value**Returns the index for which *n*-th RGBA value should be set.

sceGpIndexRgba#

Get RGBA index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**unsigned int sceGpIndexRgba#(****unsigned int *n*)**

Relative position of RGBA value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the *n*-th RGBA value should be set.**Notes**

'#' should be replaced with a string corresponding to the subtype.

This includes the following functions.

sceGpIndexRgbaLineF(), sceGpIndexRgbaLineFTS()
 sceGpIndexRgbaLineFTU()
 sceGpIndexRgbaLineG(), sceGpIndexRgbaLineGTS(),
 sceGpIndexRgbaLineGTU()
 sceGpIndexRgbaLineStripF(), sceGpIndexRgbaLineStripFTS()
 sceGpIndexRgbaLineStripFTU(), sceGpIndexRgbaLineStripG()
 sceGpIndexRgbaLineStripGTS(), sceGpIndexRgbaLineStripGTU()
 sceGpIndexRgbaPointF(), sceGpIndexRgbaPointFTS()
 sceGpIndexRgbaPointFTU()
 sceGpIndexRgbaSpriteF(), sceGpIndexRgbaSpriteFTS()
 sceGpIndexRgbaSpriteFTU()
 sceGpIndexRgbaTriF(), sceGpIndexRgbaTriFTS()
 sceGpIndexRgbaTriFTU()
 sceGpIndexRgbaTriFanF(), sceGpIndexRgbaTriFanFTS()
 sceGpIndexRgbaTriFanFTU()
 sceGpIndexRgbaTriFanG(), sceGpIndexRgbaTriFanGTS()
 sceGpIndexRgbaTriFanGTU()
 sceGpIndexRgbaTriG(), sceGpIndexRgbaTriGTS()
 sceGpIndexRgbaTriGTU()
 sceGpIndexRgbaTriStripF(), sceGpIndexRgbaTriStripFTS()

```
sceGpIndexRgbaTriStripFTU()  
sceGpIndexRgbaTriStripG(), sceGpIndexRgbaTriStripGTS()  
sceGpIndexRgbaTriStripGTU()
```

Return value

Returns the index for which n -th RGBA value should be set.

sceGpIndexSt#_R

Get ST index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
unsigned int sceGpIndexSt#_R(
    unsigned int n)
```

Relative position of ST value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the *n*-th ST value should be set.**Notes**

'#' should be replaced with a string corresponding to the subtype.

This includes the following functions.

```
sceGpIndexStLineFTS_R(), sceGpIndexStLineGTS_R()
sceGpIndexStLineStripFTS_R(), sceGpIndexStLineStripGTS_R()
sceGpIndexStPointFTS_R(), sceGpIndexStSpriteFTS_R()
sceGpIndexStTriFTS_R()
sceGpIndexStTriFanFTS_R(), sceGpIndexStTriFanGTS_R()
sceGpIndexStTriGTS_R()
sceGpIndexStTriStripFTS_R(), sceGpIndexStTriStripGTS_R()
```

Return valueReturns the index for which *n*-th ST value should be set.

sceGpIndexSt_R

Get ST index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**unsigned int sceGpIndexSt_R(**

unsigned int <i>type</i> ,	Packet type
unsigned int <i>n</i>)	Relative position of ST value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the n -th ST value should be set for an R-format drawing packet.**Return value**Returns the index for which n -th ST value should be set.

sceGpIndexStq#_P

Get STQ index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
unsigned int sceGpIndexStq#_P(
    unsigned int n)
```

Relative position of STQ value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the n -th STQ value should be set for a P-format drawing packet.**Notes**`'#'` should be replaced with a string corresponding to the subtype.

This includes the following functions.

```
sceGpIndexStqLineFTS_P(), sceGpIndexStqLineGTS_P()
sceGpIndexStqLineStripFTS_P(), sceGpIndexStqLineStripGTS_P()
sceGpIndexStqPointFTS_P(), sceGpIndexStqSpriteFTS_P()
sceGpIndexStqTriFTS_P(), sceGpIndexStqTriFanFTS_P()
sceGpIndexStqTriFanGTS_P(), sceGpIndexStqTriGTS_P()
sceGpIndexStqTriStripFTS_P(), sceGpIndexStqTriStripGTS_P()
```

Return valueReturns the index for which n -th STQ value should be set.

sceGpIndexStq_P

Get STQ index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**unsigned int sceGpIndexStq_P(**

unsigned int *type*, Packet type
unsigned int *n*) Relative position of STQ value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the *n*-th STQ value should be set for a P-format drawing packet.**Return value**Returns the index for which *n*-th STQ value should be set.

sceGpIndexUv

Get UV index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

unsigned int sceGpIndexUv(

unsigned int <i>type</i> ,	Packet type
unsigned int <i>n</i>)	Relative position of UV value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function returns the index for which the *n*-th UV value should be set.

Return value

Returns the index for which *n*-th UV value should be set.

sceGpIndexUv#

Get UV index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**unsigned int sceGpIndexUv#(**
unsigned int *n*)
Relative position of UV value
Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the *n*-th UV value should be set.**Notes**

'#' should be replaced with a string corresponding to the subtype.

This includes the following functions.

sceGpIndexUvLineFMTU(), sceGpIndexUvLineFTU()
 sceGpIndexUvLineGTU()
 sceGpIndexUvLineStripFMTU(), sceGpIndexUvLineStripFTU()
 sceGpIndexUvLineStripGTU()
 sceGpIndexUvPointFMTU(), sceGpIndexUvPointFTU()
 sceGpIndexUvSpriteFMTU(), sceGpIndexUvSpriteFTU()
 sceGpIndexUvTriFMTU(), sceGpIndexUvTriFTU()
 sceGpIndexUvTriFanFMTU(), sceGpIndexUvTriFanFTU()
 sceGpIndexUvTriFanGTU(), sceGpIndexUvTriGTU()
 sceGpIndexUvTriStripFMTU(), sceGpIndexUvTriStripFTU()
 sceGpIndexUvTriStripGTU()

Return valueReturns the index for which *n*-th UV value should be set.

sceGpIndexXyzf

Get XYZF index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

unsigned int sceGpIndexXyzf(

unsigned int <i>type</i> ,	Packet type
unsigned int <i>n</i>)	Relative position of XYZF value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function returns the index for which the *n*-th XYZF value should be set.

Return value

Returns the index for which *n*-th XYZF value should be set.

sceGpIndexXyzf#

Get XYZF index

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**unsigned int sceGpIndexXyzf#(****unsigned int *n*)**

Relative position of XYZF value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

DescriptionThis function returns the index for which the *n*-th XYZF value should be set.**Notes**

'#' should be replaced with a string corresponding to the subtype.

This includes the following functions.

sceGpIndexXyzfLineF(), sceGpIndexXyzfLineFM()
 sceGpIndexXyzfLineFMTU(), sceGpIndexXyzfLineFTS()
 sceGpIndexXyzfLineFTU()
 sceGpIndexXyzfLineG(), sceGpIndexXyzfLineGTS()
 sceGpIndexXyzfLineGTU()
 sceGpIndexXyzfLineStripF(), sceGpIndexXyzfLineStripFM()
 sceGpIndexXyzfLineStripFMTU(), sceGpIndexXyzfLineStripFTS()
 sceGpIndexXyzfLineStripFTU(), sceGpIndexXyzfLineStripG()
 sceGpIndexXyzfLineStripGTS(), sceGpIndexXyzfLineStripGTU()
 sceGpIndexXyzfPointF(), sceGpIndexXyzfPointFM()
 sceGpIndexXyzfPointFMTU(), sceGpIndexXyzfPointFTS()
 sceGpIndexXyzfPointFTU()
 sceGpIndexXyzfSpriteF(), sceGpIndexXyzfSpriteFM()
 sceGpIndexXyzfSpriteFMTU(), sceGpIndexXyzfSpriteFTS()
 sceGpIndexXyzfSpriteFTU()
 sceGpIndexXyzfTriF(), sceGpIndexXyzfTriFM()
 sceGpIndexXyzfTriFMTU(), sceGpIndexXyzfTriFTS()
 sceGpIndexXyzfTriFTU()
 sceGpIndexXyzfTriFanF(), sceGpIndexXyzfTriFanFM()
 sceGpIndexXyzfTriFanFMTU(), sceGpIndexXyzfTriFanFTS()

sceGpIndexXyzfTriFanFTU()
sceGpIndexXyzfTriFanG(), sceGpIndexXyzfTriFanGTS()
sceGpIndexXyzfTriFanGTU()
sceGpIndexXyzfTriG(), sceGpIndexXyzfTriGTS()
sceGpIndexXyzfTriGTU()
sceGpIndexXyzfTriStripF(), sceGpIndexXyzfTriStripFM()
sceGpIndexXyzfTriStripFMTU(), sceGpIndexXyzfTriStripFTS()
sceGpIndexXyzfTriStripFTU()
sceGpIndexXyzfTriStripG(), sceGpIndexXyzfTriStripGTS()
sceGpIndexXyzfTriStripGTU()

Return value

Returns the index for which n -th XYZF value should be set.

Set Functions (For Drawing Packets, Some Are Macro Functions)

sceGpSetAa1

Set AA1 ON or OFF (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
#define sceGpSetAa1(
    p,                      Pointer to packet for which value is to be set
    v)                      AA1 bit value (0: AA1 ON, 1: AA1 OFF)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the AA1 value of the specified drawing packet.

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Example:

```
u_long pPacket[PACKET_SIZE];
sceGpInitPacket(pPacket, SCE_GP_PRIM_R|SCE_GP_PRIM_SPRITE_FTU, pnum);
sceGpSetAa1((sceGpPrimR *)pPacket, 1);
```

Return value

(macro function)

sceGpSetAbe

Set alpha blending ON or OFF (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

#define sceGpSetAbe(

<i>p</i> ,	Pointer to packet for which value is to be set
<i>v</i>)	ABE bit value
	0: Alpha blending OFF
	1: Alpha blending ON

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the ABE value of the specified drawing packet.

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Example:

```

u_long pPacket[PACKET_SIZE];
sceGpInitPacket(pPacket, SCE_GP_PRIM_R|SCE_GP_PRIM_SPRITE_FTU, pnum);
sceGpSetAbe((sceGpPrimR *)pPacket, 1);

```

Return value

(macro function)

sceGpSetCtxt

Set CTEXT value (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
#define sceGpSetCtxt(
```

p ,	Pointer to packet for which value is to be set
v)	Context used (0: CTXT1, 1: CTXT2)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the CTXT value of the specified drawing packet.

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Example:

```
u_long pPacket[PACKET_SIZE];
sceGpInitPacket(pPacket, SCE_GP_PRIM_R|SCE_GP_PRIM_SPRITE_FTU, pnum);
sceGpSetCtxt((sceGpPrimR *)pPacket, 1);
```

Return value

(macro function)

sceGpSetFog

Set FOG ON or OFF (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

#define sceGpSetFog(

<i>p</i> ,	Pointer to packet for which value is to be set
<i>v</i>)	FGE bit value
	0: Fogging OFF
	1: Fogging ON

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the FGE value of the specified drawing packet.

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Example:

```

u_long pPacket[PACKET_SIZE];
sceGpInitPacket(pPacket, SCE_GP_PRIM_R|SCE_GP_PRIM_SPRITE_FTU, pnum);
sceGpSetFog((sceGpPrimR *)pPacket, 1);

```

Return value

(macro function)

sceGpSetRgb

Set RGB value (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**#define sceGpSetRgb(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>r</i> ,	R value
<i>g</i> ,	G value
<i>b</i>)	B value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the RGB value of the specified drawing packet.

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Return value

(macro function)

sceGpSetRgba

Set RGBA value (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**#define sceGpSetRgba(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>r</i> ,	R value
<i>g</i> ,	G value
<i>b</i> ,	B value
<i>a</i>)	A value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the RGBA value of the specified drawing packet.

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Return value

(macro function)

sceGpSetRgbaFM

Set color for monochrome packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**void sceGpSetRgbaFM(**

void* <i>p</i> ,	Pointer to packet for which value is to be set
u_long <i>r</i> ,	R value
u_long <i>g</i> ,	G value
u_long <i>b</i> ,	B value
u_long <i>a</i>)	A value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the color for a monochrome packet.

Notes

The userreg.DATA member of the monochrome packet structure is set to the RGBA value, and userreg.ADDR is set to SCE_GS_RGBAQ.

Q is set to 1.0f.

Return value

None

sceGpSetStq_P

Set STQ value_P (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**#define sceGpSetStq_P(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>s</i> ,	S value
<i>t</i> ,	T value
<i>q</i> ,	Q value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the STQ value at the index position of the specified drawing packet (P-format).

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Return value

(macro function)

sceGpSetStq_R

Set STQ value_R (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**#define sceGpSetStq_R(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	ST index
<i>s</i> ,	S value
<i>t</i> ,	T value
<i>q</i>)	Q value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the STQ value at the index position of the specified drawing packet (R-format).

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

With an R-format drawing packet, ST and Q are located at different index positions. However, when this function assigns the ST index, the corresponding Q will also be set at the correct position.

Return value

(macro function)

sceGpSetUv

Set UV value (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**#define sceGpSetUv(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>u</i> ,	U value
<i>v</i>)	V value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the UV value at the index position of the specified drawing packet.

Since the UV value is transferred directly to the GS and GIF, it has a fixed-point format with a 4-bit fractional part. To convert an ordinary value to this format, multiply the original value by 16.

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Return value

(macro function)

sceGpSetXy

Set XY value (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**#define sceGpSetXy(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>x</i> ,	X value
<i>y</i>)	Y value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the XY value at the index position of the specified drawing packet.

Since the XY value is transferred directly to the GS and GIF, it has a fixed-point format with a 4-bit fractional part. To convert an ordinary value to this format, multiply the original value by 16. For details, refer to the GIF Manual (for P-format) or the GS Manual (for R-format).

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

If you want to use the XYZ2 register instead of the XYZF2 register, you must modify the giftag2 member of the drawing packet. No interface is currently provided for this.

Return value

(macro function)

sceGpSetXyz

Set XYZ value (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

#define sceGpSetXyz(

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>x</i> ,	X value
<i>y</i> ,	Y value
<i>z</i>)	Z value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the XYZ value at the index position of the specified drawing packet.

Since the XY value is transferred directly to the GS and GIF, it has a fixed-point format with a 4-bit fractional part. To convert an ordinary value to this format, multiply the original value by 16.

For a P-format drawing packet, how Z is handled depends on the register that the GS will use. With normal settings (when the XYZF2 register is used), the low-order 4 bits are ignored, and the high-order bits form an unsigned integer. In this case as well, this is normally obtained by multiplying the original value by 16.

When the XYZ register is to be used, all 32 bits for Z are valid as an unsigned integer.

For details, refer to the GIF Manual (for P-format) or the GS Manual (for R-format).

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

If you want to use the XYZ2 register instead of the XYZF2 register, you must modify the giftag2 member of the drawing packet. No interface is currently provided for this.

Return value

(macro function)

sceGpSetXyz32

Set XYZ value to XYZF2 (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.5	May 10, 2002

Syntax**#define sceGpSetXyz32(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>x</i> ,	X value
<i>y</i> ,	Y value
<i>z</i> ,)	Z value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the XYZ value at the index position of the specified drawing packet.

Since the XY value is transferred directly to the GS and GIF, it has a fixed-point format with a 4-bit fractional part. To convert an ordinary value to this format, multiply the original value by 16.

All 32 bits of Z are handled as an unsigned integer.

For details, refer to the GIF Manual (for P-format) or the GS Manual (for R-format).

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

Return value

(macro function)

sceGpSetXyzf

Set XYZF value (macro function)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax**#define sceGpSetXyzf(**

<i>p</i> ,	Pointer to packet for which value is to be set
<i>k</i> ,	Index
<i>x</i> ,	X value
<i>y</i> ,	Y value
<i>z</i> ,	Z value
<i>f</i>)	F value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets the XYZF value at the index position of the specified drawing packet.

Since the XY value is transferred directly to the GS and GIF, it has a fixed-point format with a 4-bit fractional part. To convert an ordinary value to this format, multiply the original value by 16.

For a P-format drawing packet, how Z is handled depends on the register that the GS will use. With normal settings (when the XYZF2 register is used), the low-order 4 bits are ignored, and the high-order bits form an unsigned integer. In this case as well, this is normally obtained by multiplying the original value by 16.

For details, refer to the GIF Manual (for P-format) or the GS Manual (for R-format).

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

If you want to use the XYZ2 register instead of the XYZF2 register, you must modify the giftag2 member of the drawing packet. No interface is currently provided for this.

Return value

(macro function)

sceGpSetZ32

Set whether a drawing packet will use XYZF2 or XYZ2 registers

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.1	March 29, 2002

Syntax

```
void sceGpSetZ32(
```

void *p,	Pointer to packet to be set
int on)	In the drawing packet
	0: use XYZF2
	1: use XYZ2

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function sets whether the vertex coordinates of the specified drawing packet will be output to XYZF2 or XYZ2.

Notes

Since this is a macro function, the packet address must be cast to an appropriate type before it is passed.

This function overwrites the register descriptor in a GIF tag. You should prepare data corresponding to each descriptor using functions like `sceGpSetXyzf()` or `sceGpSetXyz32()`.

Example:

```
u_long pPacket[PACKET_SIZE];
sceGpInitPacket(pPacket, SCE_GP_PRIM_R|SCE_GP_PRIM_SPRITE_FTU, pnum);
sceGpSetZ32((sceGpPrimR *)pPacket, 1);
```

Return value

None

Individual Packet Transfer Functions

sceGpKickPacket

Transfer individual packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

```
int sceGpKickPacket(
    void* p,                Pointer to packet structure
    int path);              Path used
                           SCE_GP_PATH1: path1
                           SCE_GP_PATH2: path2
                           SCE_GP_PATH3: path3
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function confirms that the DMA channel is available and starts the DMA transfer of an individual packet. It returns without waiting for the end of transfer.

A function such as sceGpSetPacketMode() must be used in advance to set the packet to be transferred to a state in which an individual transfer can be performed.

When PATH1 or PATH2 is specified, DMA channel 1 is used with TTE=1 (Tag Transfer Enable ON).

When PATH3 is specified, DMA channel 2 is used with TTE=0 (Tag Transfer Enable OFF).

When PATH1 and PATH 2 packets are mixed, either path can be specified.

The end of transfer can be detected by using the sceGsSyncPath() function, for example.

This function should be called after flushing the D cache, if necessary.

Return value

Table 1-5

Value	Result
0	Transfer has started
-1	Processing for starting the transfer failed because the DMA channel used for the transfer was busy

sceGpKickPacket2

Transfer individual packet

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax

void sceGpKickPacket2(

void* *p*,

Pointer to packet structure

int *path*);

Path used

SCE_GP_PATH1: path1

SCE_GP_PATH2: path2

SCE_GP_PATH3: path3

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function starts the DMA transfer of an individual packet without confirming that the DMA channel is available. It returns without waiting for the end of transfer.

This function should be called after confirming beforehand that a DMA channel is available.

A function such as `sceGpSetPacketMode()` must be used in advance to set the packet to be transferred to a state in which an individual transfer can be performed.

When PATH1 or PATH2 is specified, DMA channel 1 is used with TTE=1 (Tag Transfer Enable ON).

When PATH3 is specified, DMA channel 2 is used with TTE=0 (Tag Transfer Enable OFF).

When PATH1 and PATH 2 packets are mixed, either path can be specified.

The end of transfer can be detected by using the `sceGsSyncPath()` function, for example.

This function should be called after flushing the D cache, if necessary.

Return value

None

sceGpSetPacketMode

Change packet transfer mode

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	July 8, 2002

Syntax

```
void sceGpSetPacketMode(
```

```
    void* p,
```

Pointer to packet

```
    int mode);
```

Transfer mode

0: Chain transfer mode

1: Individual transfer mode

2: CALL mode

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt enabled or disabled state)

Description

This function changes the transfer mode of a packet.

The packet transfer mode is 0 (chain transfer mode) after initialization.

This is the required mode when packets are added in a chain and transferred. In this case, the DMA tag at the tail position will become the NEXT tag.

To transfer a packet individually, you must set the mode to 0 (individual transfer mode). In this case, the DMA tag at the tail position will become the END tag.

In addition, in CALL mode, the DMA tag at the tail position will become the RET tag, and the call can be made externally using the CALL tag. This is used, for example, when adding a libgp packet for the HiG DMA packet management service using the sceHiDMAMake_CallPtr() function.

Return value

None

sceGpSyncPacket / sceGpSyncPacketI

Flush packet area cache

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgp	2.4.3	March 29, 2002

Syntax**void sceGpSyncPacket(**

void* p);	Starting address of packet
------------------	----------------------------

void sceGpSyncPacketI(

void* p);	Starting address of packet
------------------	----------------------------

Calling conditions

sceGpSyncPacket	Cannot be called from an interrupt handler
	Can be called from a thread
	Multithread safe (does not depend on interrupt enabled or disabled state)
sceGpSyncPacketI	Can be called from an interrupt handler
	Cannot be called from a thread

Description

This function calls the SyncDCache() / iSyncDCache() function to flush the packet area cache.

Return value

None

Chapter 2: GS Basic Library

Table of Contents

Structures	2-3
sceGsAlphaEnv	2-3
sceGsAlphaEnv2	2-4
sceGsClear	2-5
sceGsDBuff	2-6
sceGsDBuffDc	2-7
sceGsDispEnv	2-8
sceGsDrawEnv1	2-9
sceGsDrawEnv2	2-10
sceGsGParam	2-11
sceGsLoadImage	2-12
sceGsStoreImage	2-13
sceGsTexEnv	2-14
sceGsTexEnv2	2-15
Functions	2-16
sceGsExecLoadImage	2-16
sceGsExecStoreImage	2-17
sceGsGetGParam	2-18
sceGsGetIMR/isceGsGetIMR	2-19
sceGsPutDispEnv	2-20
sceGsPutDrawEnv	2-21
sceGsPutIMR/isceGsPutIMR	2-22
sceGsResetGraph	2-23
sceGsResetPath	2-24
sceGsSetDefAlphaEnv	2-25
sceGsSetDefAlphaEnv2	2-26
sceGsSetDefClear	2-27
sceGsSetDefClear2	2-29
sceGsSetDefDBuff	2-31
sceGsSetDefDBuffDc	2-33
sceGsSetDefDispEnv	2-35
sceGsSetDefDrawEnv	2-37
sceGsSetDefDrawEnv2	2-39
sceGsSetDefLoadImage	2-41
sceGsSetDefStoreImage	2-43
sceGsSetDefTexEnv	2-45
sceGsSetDefTexEnv2	2-47
sceGsSetHalfOffset	2-50
sceGsSetHalfOffset2	2-51
sceGsSwapDBuff	2-52
sceGsSwapDBuffDc	2-53
sceGsSyncPath	2-54
sceGsSyncV	2-56
sceGsSyncVCallback	2-57

Structures

sceGsAlphaEnv

Settings related to alpha blending

Library	Introduced	Documentation last modified
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGsAlpha alpha1;           ALPHA_1 register value
    long alpha1addr;             ALPHA_1 register address
    sceGsPabe pabe;              PABE register value
    long pabeaddr;               PABE register address
    sceGsTexa texa;              TEXA register value
    long texaaddr;               TEXA register address
    sceGsFba fba1;               FBA_1 register value
    long fba1addr;               FBA_1 register address
} sceGsAlphaEnv __attribute__((aligned(16)));
```

Description

This structure holds alpha blending information for context 1.

The function sceGsSetDefAlphaEnv() can be used to load values into the structure.

When a GIFtag (PACKET mode, REGS=A+D) is placed in memory immediately before this structure, information can be transferred directly to the GIF.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

sceGsAlphaEnv2

Settings related to alpha blending

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGsAlpha alpha2;           ALPHA_2 register value
    long alpha2addr;              ALPHA_2 register address
    sceGsPabe pabe;               PABE register value
    long pabeaddr;                PABE register address
    sceGsTexa texa;               TEXA register value
    long texaaddr;                TEXA register address
    sceGsFba fba2;                FBA_2 register value
    long fba2addr;                FBA_2 register address
} sceGsAlphaEnv2 __attribute__((aligned(16)));
```

Description

This structure holds alpha blending information for context 2.

The function `sceGsSetDefAlphaEnv2()` can be used to load values into the structure.

When a GIFtag (PACKET mode, REGS=A+D) is placed in memory immediately before this structure, information can be transferred directly to the GIF.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary. PABE and TEXA cannot be switched with the context. Be careful if the values are different from context 1.

sceGsClear

Data for clearing buffers

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGsTest testa;           Specified clear value of TEST register
    long testaaddr;           TEST register address
    sceGsPrim prim;           Specified clear value of sprite PRIM
    long primaddr;           PRIM register address
    sceGsRgbaq rgbaq;         Frame buffer clear value
    long rgbaqaddr;          RGBAQ register address
    sceGsXyz xyz2a;           Upper-left coordinate of sprite for clear
    long xyz2aaddr;          XYZ2 register address
    sceGsXyz xyz2b;           Lower-right coordinate of sprite for clear
    long xyz2baddr;
    sceGsTest testb;          Reset value of TEST register
    long testbaddr;          TEST register address
} sceGsClear __attribute__((aligned(16)));
```

Description

This structure holds data used for buffer clears.

When a GIFtag (PACKET mode, REGS=A+D) is placed in memory immediately before this structure, information can be transferred directly to the GIF.

The main application of this structure is for drawing simple sprites, which will be affected by immediately preceding drawing environment settings such as XYOFFSET and SCISSOR.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

sceGsDBuff

Settings for double buffering

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGsDispEnv disp[2];           Display environment
    sceGifTag giftag0;             Display environment GIFtag (for buffer 0)
    sceGsDrawEnv1 draw0;           Drawing environment (for buffer 0)
    sceGsClear clear0;             Drawing buffer clear environment (for buffer 0)
    sceGifTag giftag1;             Drawing environment GIFtag (for buffer 1)
    sceGsDrawEnv1 draw1;           Drawing environment (for buffer 1)
    sceGsClear clear1;             Drawing buffer clear environment (for buffer 1)
} sceGsDBuff;
```

Description

This structure holds data used for double buffering.

The function `sceGsSetDefDBuff()` can be used to load values into the structure.

The function `sceGsSwapDBuff()` can be used to transfer the structure's values to the GS(GIF).

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary. This structure only saves settings for context 1. Use `sceGsDbuffDc` for context 2.

sceGsDBuffDc

Settings for double buffering

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGsDispEnv disp[2];           Display environment
    sceGifTag giftag0;             Display environment GIFtag (for buffer 0)
    sceGsDrawEnv1 draw01;          Drawing environment (for context 1, buffer 0)
    sceGsDrawEnv2 draw02;          Drawing environment (for context 2, buffer 0)
    sceGsClear clear0;             Drawing buffer clear environment (for buffer 0)
    sceGifTag giftag1;             Drawing environment GIFtag (for buffer 1)
    sceGsDrawEnv1 draw11;          Drawing environment (for context 1, buffer 1)
    sceGsDrawEnv2 draw12;          Drawing environment (for context 2, buffer 1)
    sceGsClear clear1;             Drawing buffer clear environment (for buffer 1)
} sceGsDBuffDc;
```

Description

This structure holds data used for double buffering, for context 2.

The function `sceGsSetDefDBuffDc()` can be used to load values into the structure.

The function `sceGsSwapDBuffDc()` can be used to transfer the structure's values to the GS(GIF).

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

sceGsDispEnv

Settings for the display environment

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	March 26, 2001

Structure

```
typedef struct {
    tGS_PMODE pmode;           PCRTC mode setting (value of PMODE register)
    tGS_SMODE2 smode2;         Video sync mode setting (value of SMODE2 register)
    tGS_DISPFB2 dispfb;        Display frame buffer setting (value of DISPFB2 register)
    tGS_DISPLAY2 display;       Display position setting on video screen (value of
                                DISPLAY2 register)
    tGS_BGCOLOR bgbcolor;       Background color setting (value of BGCOLOR register)
} sceGsDispEnv;
```

Description

This structure holds display-related settings, GS rectangle read data, and settings related to circuit 2.

The function `sceGsSetDefDispEnv()` can be used to load values into the structure.

The function `sceGsPutDispEnv()` can be used to transfer the structure's values to the GS.

sceGsDrawEnv1

Settings for the drawing environment

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGsFrame frame1;           FRAME_1 register value
    u_long frame1addr;          FRAME_1 register address
    sceGsZbuf zbuf1;            ZBUF_1 register value
    long zbuf1addr;             ZBUF_1 register address
    sceGsXyoffset xyoffset1;    XYOFFSET_1 register value
    long xyoffset1addr;         XYOFFSET_1 register address
    sceGsScissor scissor1;      SCISSOR_1 register value
    long scissor1addr;          SCISSOR_1 register address
    sceGsPrmodecont prmodecont; PRMODECONT register value
    long prmodecontaddr;        PRMODECONT register address
    sceGsColclamp colclamp;     COLCLAMP register value
    long colclampaddr;          COLCLAMP register address
    sceGsDthe dthe;             DTHE register value
    long dtheaddr;              DTHE register address
    sceGsTest test1;            TEST_1 register value
    long test1addr;             TEST_1 register address
} sceGsDrawEnv1 __attribute__((aligned(16)));
```

Description

This structure holds information about the drawing environment (context 1).

The function `sceGsSetDefDrawEnv()` can be used to load values into the structure.

When a GIFTAG (PACKED mode, REGS=A+D) is placed in memory immediately before this structure, the function `sceGsPutDrawEnv()` can be used to transfer information directly to the GS.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

sceGsDrawEnv2

Settings for the drawing environment

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGsFrame frame2;           FRAME_2 register value
    u_long frame2addr;          FRAME_2 register address
    sceGsZbuf zbuf2;           ZBUF_2 register value
    long zbuf2addr;            ZBUF_2 register address
    sceGsXyoffset xyoffset2;    XYOFFSET_2 register value
    long xyoffset2addr;        XYOFFSET_2 register address
    sceGsScissor scissor2;     SCISSOR_2 register value
    long scissor2addr;         SCISSOR_2 register address
    sceGsPrmodecont prmodecont; PRMODECONT register value
    long prmodecontaddr;      PRMODECONT register address
    sceGsColclamp colclamp;    COLCLAMP register value
    long colclampaddr;        COLCLAMP register address
    sceGsDthe dthe;           DTHE register value
    long dtheaddr;           DTHE register address
    sceGsTest test2;          TEST_2 register value
    long test2addr;          TEST_2 register address
} sceGsDrawEnv2 __attribute__((aligned(16)));
```

Description

This structure holds information about the drawing environment (context 2).

The function `sceGsSetDefDrawEnv2()` can be used to load values into the structure.

When a GIFtag (PACKED mode, REGS=A+D) is placed in memory immediately before this structure, the function `sceGsPutDrawEnv()` can be used to transfer information directly to the GS.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

PRMODECONT, COLCLAMP and DTHE cannot be switched with the context. Be careful if the values are different from context 1.

sceGsGParam

Library system information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.6	March 26, 2001

Structure

```
typedef struct {
    short  sceGsInterMode;           Interlace/non-interlace value
    short  sceGsOutMode;             NTSC/PAL value
    short  sceGsFFMode;              FIELD/FRAME value
    short  sceGsVersion;             GS version
    volatile int (*sceGsVSCfunc)(int); Callback function pointer set by sceGsSyncVCallback
    int  sceGsVSCid;                 Interrupt handler ID
} sceGsGParam__attribute__((aligned(16)));
```

Description

This structure holds the parameters used by the library.

Data being used by the library can be referenced using sceGsGetGParam().

sceGsLoadImage

Data structure for LoadImage

Library	Introduced	Documentation last modified
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    sceGifTag giftag0;           GIFtag for data transfer settings
    sceGsBitbltbuf bitbltbuf;    BITBLTBUF register value
    long bitbltbufaddr;          BITBLTBUF register address
    sceGsTrxpos trxpos;          TRXPOS register value
    long trxposaddr;             TRXPOS register address
    sceGsTrxreg trxreg;          TRXREG register value
    long trxregaddr;             TRXREG register address
    sceGsTrxdir trxdir;          TRXDIR register value
    long trxdiraddr;             TRXDIR register address
    sceGifTag giftag1;           GIFtag for image transfer
} sceGsLoadImage __attribute__((aligned(16)));
```

Description

This structure is used for transferring image data to the GS.

The function sceGsSetDefLoadImage() can be used to load values into the structure.

This structure can be sent directly to the GIF, followed by the image data.

A simpler method involves transferring image data to the GS using the function sceGsExecLoadImage().

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

sceGsStoreImage

Data structure for StoreImage

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 23, 1999

Structure

```
typedef struct {
    u_int vifcode[4];           Path2 VIFcode value
    sceGifTag giftag;           GIFtag for data transfer
    sceGsBitbltbuf bitbltbuf;   BITBLTBUF register value
    long bitbltbufaddr;         BITBLTBUF register address
    sceGsTrxpos trxpos;         TRXPOS register value
    long trxposaddr;            TRXPOS register address
    sceGsTrxreg trxreg;         TRXREG register value
    long trxregaddr;            TRXREG register address
    sceGsFinish finish;         FINISH register value
    long finishaddr;            FINISH register address
    sceGsTrxdir trxdir;         TRXDIR register value
    long trxdiraddr;            TRXDIR register address
} sceGsStoreImage __attribute__((aligned(16)));
```

Description

This structure is used for transferring image data to the GS via PATH2.

The function `sceGsSetDefStoreImage()` can be used to load values into the structure.

A simpler method involves transferring image data to the GS using the function `sceGsExecStoreImage()`.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

sceGsTexEnv

Settings related to textures

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	March 31, 2000

Structure

```
typedef struct {
    sceGsTexflush texflush;           TEXFLUSH register value
    long texflushaddr;                TEXFLUSH register address
    sceGsTex1 tex11;                  TEX1_1 register value
    long tex11addr;                    TEX1_1 register address
    sceGsTex0 tex01;                  TEX0_1 register value
    long tex01addr;                    TEX0_1 register address
    sceGsClamp clamp1;                CLAMP_1 register value
    long clamp1addr;                  CLAMP_1 register address
} sceGsTexEnv __attribute__((aligned(16)));
```

Description

This structure holds texture information for context 1.

The function sceGsSetDefTexEnv() can be used to load values into the structure.

When a GIFtag (PACKET mode, REGS=A+D) is placed in memory immediately before this structure, information can be transferred directly to the GIF.

When mipmap is used, the MIPTBP1_1 and MIPTBP2_1 registers must also be set.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

sceGsTexEnv2

Settings related to textures

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	March 31, 2000

Structure

```
typedef struct {
    sceGsTexflush texflush;           TEXFLUSH register value
    long texflushaddr;                 TEXFLUSH register address
    sceGsTex1 tex12;                   TEX1_2 register value
    long tex12addr;                     TEX1_2 register address
    sceGsTex0 tex02;                   TEX0_2 register value
    long tex02addr;                     TEX0_2 register address
    sceGsClamp clamp2;                 CLAMP_2 register value
    long clamp2addr;                   CLAMP_2 register address
} sceGsTexEnv2 __attribute__((aligned(16)));
```

Description

This structure holds texture information for context 2.

The function `sceGsSetDefTexEnv2()` can be used to load values into the structure.

When a GIFtag (PACKET mode, REGS=A+D) is placed in memory immediately before this structure, information can be transferred directly to the GIF.

When mipmap is used, the MIPTBP1_2 and MIPTBP2_2 registers must also be set.

Notes

Since the contents of this structure are transferred directly to the GS with DMA, the data must be aligned on a 16-byte boundary.

Functions

sceGsExecLoadImage

Execute LoadImage

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 6, 2002

Syntax

```
int sceGsExecLoadImage(
    sceGsLoadImage *lp,           Address of structure containing LoadImage information
    u_long128 *srcaddr)          Data transfer source address
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function transfers image data from GS main memory to GS local memory via PATH3. The information needed for the transfer must be set up in the sceGsLoadImage structure beforehand using the sceGsSetDefLoadImage() function.

Because this is a simplified version provided for prototyping and debugging, the DMA channel (ch-2) must be idle, otherwise program execution will be blocked.

Notes

If the sceGsLoadImage structure is specified as being cached, the D-cache must be flushed back to memory before calling this function. When a new texture is transferred to the GS, setup must be performed again for the texture.

Please refer to the description of the sceGsSetDefTexEnv() function.

Return value

Returns the termination state

When termination is normal, returns 0.

When there is a timeout, returns -1.

sceGsExecStoreImage

Execute StoreImage

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

```
int sceGsExecStoreImage(
    sceGsStoreImage *sp,           Address of structure containing StoreImage information
    u_long128 *dstaddr)           Data transfer destination address
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function transfers image data from GS local memory to GS main memory.

The information needed for the transfer must be set up in the sceGsStoreImage structure beforehand using the sceGsSetDefStoreImage() function.

Because this is a simplified version provided for prototyping and debugging, the DMA channel (ch-2) must be idle, otherwise program execution will be blocked.

Since the datapath used for the transfer is PATH2, other datapaths will also be inhibited.

Notes

The DMA environment is not saved in the sceGsStoreImage() structure.

As a result, it is initialized to DI_CHCR.TTE=0 immediately after execution.

When Cached is specified for the sceGsStoreImage structure, be sure to flush the D-cache and perform a writeback to memory before this function is called.

Return value

Returns the termination state.

When termination is normal, returns 0.

When there is a timeout, returns -1.

sceGsGetGParam

Get library system information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.6	December 6, 2002

Syntax

sceGsGParam *sceGsGetGParam(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

Returns a pointer to the system information structure currently being used by the library.

Return value

Returns a pointer to the structure currently being used by the system.

sceGsGetIMR/isceGsGetIMR

Get interrupt mask

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax**u_long** sceGsGetIMR(void)**u_long** isceGsGetIMR(void)**Calling conditions**

sceGsGetIMR	Cannot be called from an interrupt handler Can be called from a thread Not multithread safe (does not depend on interrupt disabled or enabled state)
isceGsGetIMR	Can be called from an interrupt handler Cannot be called from a thread

Description

Since the IMR is write-only, it isn't possible to check the current setting of the GS interrupt mask. If the IMR is written using only sceGsPutIMR, it will be possible to confirm the value previously set.

To get the interrupt mask within an interrupt handler, use the isceGsGetIMR() function.

Return value

Returns the IMR register value previously set by sceGsPutIMR/isceGsPutIMR.

sceGsPutDispEnv

Initialize the display environment

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 6, 2002

Syntax**void sceGsPutDispEnv(**
sceGsDispEnv *disp) Address of structure used to set up the display environment
Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionTransfers the contents of the structure pointed to by *disp* to the corresponding GS registers.**Return value**

None

sceGsPutDrawEnv

Initialize the drawing environment

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 6, 2002

Syntax

```
int sceGsPutDrawEnv(
    sceGifTag *giftag)           Starting address of drawing environment settings data
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function transfers drawing environment settings data to the GS and sets up the necessary registers.

A drawing environment structure `sceGsDrawEnv` holding appropriate values should be prepared and preceded immediately by a `GIFtag` (PACKED mode, REGS=A+D). The address of the `GIFtag` should be specified in the *giftag* argument.

This function terminates right after DMA transfer is begun (without waiting for the transfer to finish).

Notes

Since the data is transferred via PATH3, the GIF channel and the GIF must be idle, and PATH3 must not be masked when this function is called. If these are not in idle state, the program will be blocked until they enter idle state.

Since data transfers are performed using DMA, the `GIFtag` and subsequent data must be memory resident. Before this function is called, data must be flushed back to memory from the D-cache on the user side.

Return value

Returns the termination state.

When termination is normal, returns 0.

When there is a timeout (when preceding Ch-2 DMA has not finished), returns -1.

sceGsPutIMR/isceGsPutIMR

Set up interrupt mask

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

```

u_long sceGsPutIMR(
    u_long imr)                Value to be stored in the IMR
u_long isceGsPutIMR(
    u_long imr)                Value to be stored in the IMR

```

Calling conditions

sceGsPutIMR Cannot be called from an interrupt handler
 Can be called from a thread
 Not multithread safe (does not depend on interrupt disabled or enabled state)

isceGsPutIMR Can be called from an interrupt handler
 Cannot be called from a thread

Description

Since the IMR is write-only, it isn't possible to check the current setting of the GS interrupt mask. If the IMR is written using only this function, it will be possible to confirm the value previously set.

To set the interrupt mask within an interrupt handler, use the isceGsGetIMR() function.

Return value

Returns the IMR register value previously set by sceGsPutIMR/isceGsPutIMR.

sceGsResetGraph

Initialize the GS

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

void sceGsResetGraph(

short <i>mode</i> ,	Reset mode 0: General reset 1: Drawing reset
short <i>inter</i> ,	Interlace/non-interlace settings (valid only when mode==0) 0: Non-interlace 1: Interlace
short <i>omode</i> ,	Video signal format setting (valid only when mode==0) 0x02: NTSC 0x03: PAL 0x50: DTV480P
short <i>ffmode</i>)	FRAME/FIELD mode setting (valid only in interlace mode) 0: Read every other line beginning with start of FIELD (+0,+2,+4,... / +1,+3,+5,...) 1: Read each line beginning with start of FRAME (+0,+1,+2,+3,...)

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

The GS is reset according to the specified mode.

The entire GS is reset if *mode* = 0.

If *mode* = 1, drawing operations are canceled and the primitive data in the internal buffer of the GS is discarded (drawing environment and display environment settings are saved).

Because settings related to television screen display may differ as a function of GS chip version, this function should be used for initializing the GS, otherwise, display problems may occur.

Notes

The contents of GS local memory cannot be guaranteed during a general reset.

Additionally, callback functions registered with the sceGsSyncVCallback function are also deleted during a general reset.

After issuing sceSifRebootlop(), do not call this function until you can confirm that the IOP has restarted.

Return value

None

sceGsResetPath

Initialize datapath device

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 6, 2002

Syntax**void sceGsResetPath(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Resets the devices on the data transfer path to the GS, i.e., VIF1, VU1 and the GIF.

Notes

The general-purpose registers of VIF1 are initialized as follows:

STCYCL	(WL=4, CL=4)
STMASK	(all 0)
STMOD	(MOD=0)
MSKPATH3	(0: enable transfers)
BASE	(0)
OFFSET	(0)
ITOP	(0)

Also, ME0 of the privileged register, VIF1_ERR, is set to 1.

Return value

None

sceGsSetDefAlphaEnv

Generate alpha blending environment data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefAlphaEnv(

sceGsAlphaEnv *ap,

Address of structure used to set up alpha blending information

short pabe)

Pixel-by-pixel alpha blending

0: No

1: Yes (Alpha blending is turned off if the MSB of the A value in a pixel is 0)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads alpha blending settings information into the members of the sceGsAlphaEnv structure for context 1.

When a GIFtag (PACKED mode, REGS=A+D) is placed in memory immediately before the structure, information can be transferred directly to the GIF.

If the structure is cached, data will not be transferred properly unless the D-cache is flushed back to memory.

Notes

This function loads the following values into the members of the sceGsAlphaEnv structure. Other values can be used if necessary.

alpha1.B = 1

alpha1.D = 1

pabe = pabe

texa.TA0 = 127

texa.AEM = 1

texa.TA1 = 129

Return value

Returns the size of the sceGsAlphaEnv structure (in qwords).

sceGsSetDefAlphaEnv2

Generate alpha blending environment data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefAlphaEnv2(

sceGsAlphaEnv2 *ap,

Address of structure used to set up alpha blending information

short pabe)

Pixel-by-pixel alpha blending

0: No

1: Yes (Alpha blending is turned off if the MSB of the A value in a pixel is 0)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads alpha blending settings information into the members of the sceGsAlphaEnv structure for context 2.

When a GIFtag (PACKED mode, REGS=A+D) is placed in memory immediately before the structure, information can be transferred directly to the GIF.

If the structure is cached, data will not be transferred properly unless the D-cache is flushed back to memory.

PABE and TEXA cannot be switched with the context. Be careful if the values are different from context 1.

Notes

This function loads the following values into the members of the sceGsAlphaEnv2 structure. Other values can be used if necessary.

alpha2.B = 1

alpha2.D = 1

pabe = pabe

texa.TA0 = 127

texa.AEM = 1

texa.TA1 = 129

Return value

Returns the size of the sceGsAlphaEnv2 structure (in qwords).

sceGsSetDefClear

Generate buffer clear data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefClear(

sceGsClear *cp,

Address of the structure for setting up buffer clear

short ztest,

Depth testing method

0: no depth testing (Z-buffer not used)

1: Draw all pixels regardless of Z-buffer value

2: Draw pixels whose Z values are greater than or equal to the Z-buffer value

3: Draw pixels whose Z values are greater than the Z-buffer value

short x, **short** y,

Upper left coordinate of clear area

short w, **short** h,

Width, height of clear area (in pixels)

u_char r, **u_char** g, **u_char** b, **u_char** a,

Clear value for frame buffer

u_int z)

Clear value for Z-buffer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads buffer clear data into the members of the specified sceGsClear structure.

When a GIFtag (PACKED mode, A+D) is placed immediately before the structure and information is transferred to the GIF, the context 1 frame buffer will be cleared to the values specified by *r*, *g*, *b*, *a*, and likewise, the context 1 Z-buffer will be cleared to the value specified by *z*.

If *ztest* is set to 1-3, the depth testing method is set temporarily to ALWAYS, the frame buffer and the Z-buffer are cleared, and the method specified in *ztest* is set up.

If *ztest* is set to 0, the function performs an action equivalent to *ztest* = 1, therefore it must be used with ZMSK of ZBUF set to 1.

Notes

If *ztest* != 0, the following values will be loaded into the members of the sceGsClear structure. Values can be changed as needed.

testa.ZTE = 1

testa.ZTST = 1

prim = 6(SPRITE)

rgbaq.R = *r*

rgbaq.G = *g*

rgbaq.B = *b*

rgbaq.A = *a*

```
rgbaq.Q = 1.0f  
xyz2a.X = x<<4  
xyz2a.Y = y<<4  
xyz2a.Z = z  
xyz2b.X = (x+w)<<4  
xyz2b.Y = (y+h)<<4  
xyz2b.Z = z  
testb.ZTE = 1  
testb.ZTST = ztest
```

Return value

Returns the size of the sceGsClear structure (in qwords).

sceGsSetDefClear2

Generate buffer clear data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefClear2(

sceGsClear *cp,

Address of the structure for setting up buffer clear

short ztest,

Depth testing method

0: no depth testing (Z-buffer not used)

1: Draw all pixels regardless of Z-buffer value

2: Draw pixels whose Z values are greater than or equal to the Z-buffer value

3: Draw pixels whose Z values are greater than the Z-buffer value

short x, short y,

Upper left coordinate of clear area

short w, short h,

Width, height of clear area (in pixels)

u_char r, u_char g, u_char b, u_char a,

Clear value for frame buffer

u_int z)

Clear value for Z-buffer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads buffer clear data into the members of the specified sceGsClear structure.

When a GIFtag (PACKED mode, A+D) is placed immediately before the structure and information is transferred to the GIF, the context 2 frame buffer will be cleared to the values specified by *r*, *g*, *b*, *a*, and likewise, the context 2 Z-buffer will be cleared to the value specified by *z*.

If *ztest* is set to 1-3, the depth testing method is set temporarily to ALWAYS, the frame buffer and the Z-buffer are cleared, and the method specified in *ztest* is set up.

If *ztest* is set to 0, the function performs an action equivalent to *ztest* = 1, therefore it must be used with ZMSK of ZBUF set to 1.

Notes

If *ztest* != 0, the following values will be loaded into the members of the sceGsClear structure. Values can be changed as needed.

testa.ZTE = 1

testa.ZTST = 1

prim = 6(SPRITE)

rgbaq.R = *r*

rgbaq.G = *g*

rgbaq.B = *b*

rgbaq.A = *a*

```
rgbaq.Q = 1.0f  
xyz2a.X = x<<4  
xyz2a.Y = y<<4  
xyz2a.Z = z  
xyz2b.X = (x+w)<<4  
xyz2b.Y = (y+h)<<4  
xyz2b.Z = z  
testb.ZTE = 1  
testb.ZTST = ztest
```

Return value

Returns the size of the sceGsClear structure (in qwords).

sceGsSetDefDBuff

Set up double buffering

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

void sceGsSetDefDBuff(

sceGsDBuff *db,	Address of structure used to set up double buffering
short psm,	Draw pixel format 0: PSMCT32 1: PSMCT24 2: PSMCT16 10: PSMCT16S
short w, short h,	Width, height of display/drawing environment (in pixels)
short ztest,	Method used for depth testing 0: No depth testing (Z-buffer mask) 1: Draw all pixels regardless of Z-buffer value 2: Draw pixels whose Z values are greater than or equal to the Z-buffer value 3: Draw pixels whose Z values are greater than the Z-buffer value
short zpsm	Format in which Z values are stored (only valid when ztest!=0) 0: PSMZ32 1: PSMZ24 2: PSMZ16 10: PSMZ16S
short clear)	Clearing of drawing area 0: Do not clear 1: Clear

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt enabled state)

Description

This function uses the given arguments to load values into the two sets of drawing environment/display environment/buffer clear data structures contained in the double buffer information structure.

If 1 is specified for *clear*, the frame buffer and the Z-buffer will be cleared when the double buffers are swapped using the `sceGsSwapDBuff()` function.

This function can only be used to set the context 1 environment.

If the function is called with `ztest = 0`, `ztest = 1` `ZBUF.ZMSK=1` is set.

Notes

If *clear* =1 and *psm* = PSMCT32 are specified in interlace/Frames mode, this function will load the following member values into the structure. The values can be changed if necessary.

Table 2-1: Loaded member values

Item	Description
disp[0]	Results from sceGsSetDefDispEnv(&db->disp[0], psm, w, h, 0, 0)
disp[1]	Results from sceGsSetDefDispEnv(&db->disp[1], psm, w, h, 0, 0) where disp[1].dispfb.FBP = $((w+63)/64)*((h+31)/32)$
draw0	Results from sceGsSetDefDrawEnv(&db->draw0, psm, w, h, ztest, zpsm) where draw0.frame1.FBP = $((w+63)/64)*((h+31)/32)$
draw1	Results from sceGsSetDefDrawEnv(&db->draw1, psm, w, h, ztest, zpsm)
clear0	Results from sceGsSetDefClear(&db->clear0, ztest, 2048-(w>>1), 2048-(h>>1), w, h, 0, 0, 0, 0, 0)
clear1	Results from sceGsSetDefClear(&db->clear1, ztest, 2048-(w>>1), 2048-(h>>1), w, h, 0, 0, 0, 0, 0)

Return value

None

sceGsSetDefDBuffDc

Set up double buffering

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

```
void sceGsSetDefDBuffDc(
    sceGsDBuffDc *db,           Address of structure used to set up double buffering
    short psm,                  Draw pixel format
                                0: PSMCT32
                                1: PSMCT24
                                2: PSMCT16
                                10: PSMCT16S
    short w, short h,           Width, height of display/drawing environment (in pixels)
    short ztest,                Method used for depth testing
                                0: No depth testing (Z-buffer mask)
                                1: Draw all pixels regardless of Z-buffer value
                                2: Draw pixels whose Z values are greater than or equal
                                   to the Z-buffer value
                                3: Draw pixels whose Z values are greater than the
                                   Z-buffer value
    short zpsm                  Format in which Z values are stored (only valid when
                                ztest!=0)
                                0: PSMZ32
                                1: PSMZ24
                                2: PSMZ16
                                10: PSMZ16S
    short clear)                Clearing of drawing area
                                0: Do not clear
                                1: Clear
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt enabled state)

Description

This function uses the given arguments to load values into the two sets of drawing environment/display environment/buffer clear data structures contained in the double buffer information structure.

If 1 is specified for clear, the frame buffer and the Z-buffer will be cleared when the double buffers are swapped using the sceGsSwapDBuff() function.

This function can be used to set both the context 1 and context 2 environments.

If the function is called with ztest = 0, ztest = 1 ZBUF.ZMSK=1 is set.

Notes

If clear =1 and psm = PSMCT32 are specified in interlace/Frames mode, this function will load the following member values into the structure. The values can be changed if necessary.

Table 2-2: Loaded member values

Item	Description
disp[0]	results from sceGsSetDefDispEnv(&db->disp[0], psm, w, h, 0, 0)
disp[1]	results from sceGsSetDefDispEnv(&db->disp[1], psm, w, h, 0, 0) where disp[1].dispfb.FBP = ((w+63)/64)*((h+31)/32)
draw01	results from sceGsSetDefDrawEnv(&db->draw01, psm, w, h, ztest, zpsm) where draw01.frame1.FBP = ((w+63)/64)*((h+31)/32)
draw02	results from sceGsSetDefDrawEnv(&db->draw02, psm, w, h, ztest, zpsm) where draw02.frame1.FBP = ((w+63)/64)*((h+31)/32)
draw11	results from sceGsSetDefDrawEnv(&db->draw11, psm, w, h, ztest, zpsm)
draw12	results from sceGsSetDefDrawEnv(&db->draw12, psm, w, h, ztest, zpsm)
clear0	results from sceGsSetDefClear(&db->clear0, ztest, 2048-(w>>1), 2048-(h>>1), w, h, 0, 0, 0, 0, 0)
clear1	results from sceGsSetDefClear(&db->clear1, ztest, 2048-(w>>1), 2048-(h>>1), w, h, 0, 0, 0, 0, 0)

Return value

None

sceGsSetDefDispEnv

Generate display environment data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

```
void sceGsSetDefDispEnv(
    sceGsDispEnv *disp,           Address of structure used to set up the display
                                   environment
    short psm,                    Pixel format
                                   0: PSMCT32
                                   1: PSMCT24
                                   2: PSMCT16
                                   10: PSMCT16S
    short w, short h,             Width, height of display area (in pixels)
    short dx, short dy)           Position on TV screen of the upper left point of the display
                                   area (in pixels)
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt enabled state)

Description

This function loads values into the members of the display environment structure.

Notes

If interlace/FRAME mode is specified, this function loads the following values into the members of the structure. These values can be reset if necessary.

NTSC mode:

```
pmode = 0x66
smode2 = 3
dispfb2.PSM = psm;
dispfb2.FBW = (w>>6)<<9
display2.DH = ((h<<1)-1) << 44
display2.DW = (0x9ff)<<32
display2.MAGH = (((2560+w-1)/w)-1) << 23
display2.DY = (50+dy) <<12
display2.DX = 0x27c + (dx*(2560/w))
bgcolor = 0
```

PAL mode:

```
pmode = 0x66
smode2 = 3
dispfb.PSM = psm;
dispfb.FBW = (w>>6)<<9
display2.DH = ((h<<1)-1) << 44
display2.DW = (0x9ff)<<32
display2.MAGH = (((2560+w-1)/w)-1) << 23
display2.DY = 72+dy
display2.DX = 0x290 + (dx*(2560/w))
bgcolor = 0
```

Return value

None

sceGsSetDefDrawEnv

Generate drawing environment data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax**int sceGsSetDefDrawEnv(****sceGsDrawEnv1 *draw,**

Address of structure used to set up the drawing environment

short psm,

Draw pixel format

0: PSMCT32

1: PSMCT24

2: PSMCT16

10: PSMCT16S

short w, short h,

Width, height of drawing area (in pixels)

short ztest,

Depth testing method

0: No depth testing (Z-buffer mask)

1: Draw all pixels regardless of Z-buffer value

2: Draw pixels whose Z values are greater than or equal to the Z-buffer value

3: Draw pixels whose Z values are greater than the Z-buffer value

short zpsm)

Z-value storage format (valid only when ztest!=0)

0: PSMZ32

1: PSMZ24

2: PSMZ16

10: PSMZ16S

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads values into the members of the drawing environment structure for context 1.

Notes

When cached is specified for *draw*, the contents of *draw* will be cache-resident when this function completes. Therefore, the D-cache must be flushed back to memory before performing a DMA transfer. If the function is called with *ztest* = 0, *ztest* = 1 ZBUF.ZMSK=1 is set.

When *ztest* != 0 and *psm* = PSMCT32, the following member values are generated by this function. These values can be changed if necessary.

frame1.PSM = psm

frame1.FBW = w

zbuf1.ZBP = ((w+63)/64)*((h+31)/32)*2;

zbuf1.ZPSM = zpsm

xyoffset1.OFX = (2048 - (w>>1))<<4;

```
xyoffset1.OFY = (2048 - (h>>1))<<4;  
scissor1.SCAX1 = w-1  
scissor1.SCAY1 = h-1  
prmodecont.AC = 1  
colclamp.CLAMP = 1  
dthe.DTHE = 0  
test1.ZTE = 1  
test1.ZTST = ztest
```

Return value

Returns the size of the sceGsDrawEnv1 structure (in qwords).

sceGsSetDefDrawEnv2

Generate drawing environment data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefDrawEnv2(

sceGsDrawEnv2 *draw,	Address of structure used to set up the drawing environment
short psm,	Draw pixel format 0: PSMCT32 1: PSMCT24 2: PSMCT16 10: PSMCT16S
short w, short h,	Width, height of drawing area (in pixels)
short ztest,	Depth testing method 0: No depth testing (Z-buffer mask) 1: Draw all pixels regardless of Z-buffer value 2: Draw pixels whose Z values are greater than or equal to the Z-buffer value 3: Draw pixels whose Z values are greater than the Z-buffer value
short zpsm)	Z-value storage format (valid only when ztest!=0) 0: PSMZ32 1: PSMZ24 2: PSMZ16 10: PSMZ16S

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads values into the members of the drawing environment structure for context 2.

Notes

When cached is specified for draw, the contents of draw will be cache-resident when this function completes. Therefore, the D-cache must be flushed back to memory before performing a DMA transfer.

PRMODECONT, COLCLAMP and DTHE cannot be switched with the context. Be careful if the values are different for context 1. If the function is called with ztest = 0, ztest = 1 ZBUF.ZMSK=1 is set.

When ztest != 0 and psm = PSMCT32, the following member values are generated by this function. These values can be changed if necessary.

```
frame2.PSM = psm
frame2.FBW = w
zbuf2.ZBP = ((w+63)/64)*((h+31)/32)*2;
```

```
zbuf2.ZPSM = zpsm  
xyoffset2.OFX = (2048 - (w>>1))<<4;  
xyoffset2.OFY = (2048 - (h>>1))<<4;  
scissor2.SCAX1 = w-1  
scissor2.SCAY1 = h-1  
prmodecont.AC = 1  
colclamp.CLAMP = 1  
dthe.DTHE = 0  
test2.ZTE = 1  
test2.ZTST = ztest
```

Return value

Returns the size of the sceGsDrawEnv2 structure (in qwords).

sceGsSetDefLoadImage

Set up LoadImage information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefLoadImage(

sceGsLoadImage **p*,

Address of structure used to set up LoadImage information

short *dbp*,

Base address of destination buffer for dbp transfer
(actual address will be *dbp* x 64)

short *dbw*,

Width of dbw destination buffer
(actual width will be *dbw* x 64)

short *dpsm*,

dpsm Pixel format for data transfer
0: PSMCT32 (pixel size: 32bit)
1: PSMCT24 (pixel size: 24bit)
2: PSMCT16 (pixel size: 16bit)
10: PSMCT16S (pixel size: 16bit)
19: PSMT8 (pixel size: 8bit)
20: PSMT4 (pixel size: 4bit)
27: PSMT8H (pixel size: 8bit)
36: PSMT4HL (pixel size: 4bit)
44: PSMT4HH (pixel size: 4bit)
48: PSMZ32 (pixel size: 32bit)
49: PSMZ24 (pixel size: 24bit)
50: PSMZ16 (pixel size: 16bit)
58: PSMZ16S (pixel size: 16bit)

short *x*, **short** *y*,

Upper left coordinates for transfer destination area

short *w*, **short** *h*)

Width, height of transfer area (in pixels)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets up a structure for transferring image data to the GS via PATH3. The structure contains a GIFtag so it can be sent directly to the GS before the image data is transferred. The size of the image data (*w* x *h* x pixel size) must be a multiple of 16 bytes and must be 32767 x 16 bytes or less.

If the pixel size is 8 bits, *x* and *w* must be multiples of 2. If the pixel size is 4 bits, *x* and *w* must be multiples of 4.

Notes

This function loads the following values into the members of the sceGsLoadImage structure. These values can be changed if necessary.

bitbltbuf.DBP = *dbp*

bitbltbuf.DBW = *dbw*

```
bitbltbuf.DPSM = dpsm  
trxpos.DSAX = x  
trxpos.DSAY = y  
trxreg.RRW = w  
trxreg.RRH = h
```

The `sceGsExecLoadImage()` function can be used to easily perform data transfers using the `sceGsLoadImage` structure.

Return value

Returns the size of the `sceGsLoadImage` structure (in qwords).

sceGsSetDefStoreImage

Set up StoreImage information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefStoreImage(

sceGsStoreImage *sp,	Address of structure used to set up StoreImage information
short sbp,	Base address of transfer destination buffer (actual address will be sbp x 64)
short sbw,	Width of transfer source buffer (actual width will be sbw x 64)
short spsm,	Pixel format of transfer data 0: PSMCT32 (pixel size: 32bit) 1: PSMCT24 (pixel size: 24bit) 2: PSMCT16 (pixel size: 16bit) 10: PSMCT16S (pixel size: 16bit) 19: PSMT8 (pixel size: 8bit) 27: PSMT8H (pixel size: 8bit) 48: PSMZ32 (pixel size: 32bit) 49: PSMZ24 (pixel size: 24bit) 50: PSMZ16 (pixel size: 16bit) 58: PSMZ16S (pixel size: 16bit)
short x, short y,	Upper left coordinates for transfer source
short w, short h)	Width, height of transfer area (in pixels)

Note: Due to hardware specifications, PSMT4, PSMT4HH and PSMT4HL local to host transfers are not possible. The data must first be obtained in another mode and then rearranged.

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets up a structure for transferring image data to the GS via PATH2. The structure contains a GIFtag so it can be sent directly to the GS.

The size of the image data ($w \times h \times \text{pixel size}$) must be a multiple of 16 bytes and must be 32767 x 16 bytes or smaller.

If the pixel size is 8 bits, x and w must be multiples of 2.

Notes

This function loads the following values into the members of the sceGsStoreImage structure. These values can be changed as needed.

vifcode[0] = VIFNOP

```

vifcode[1] = VIFMSKPATH3 (MASK on)
vifcode[2] = VIFFLUSHA
vifcode[3] = DIRECT
bitbltbuf.SBP = sbp
bitbltbuf.SBW = sbw
bitbltbuf.SPSM = spsm
trxpos.SSAX = x
trxpos.SSAY = y
trxreg.RRW = w
trxreg.RRH = h
trxdir.DIR = 1

```

The `sceGsExecStoreImage()` function can be used to easily perform data transfers using the `sceGsStoreImage` structure.

Return value

Returns the size of the `sceGsStoreImage` structure (in qwords).

sceGsSetDefTexEnv

Generate texture environment settings

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax**int sceGsSetDefTexEnv(****sceGsTexEnv** **tp*,

Address of texture information settings structure

short *flush*,

Disable texture page buffer

0: Do not disable

1: Disable

short *tbp0*,

Base address of texture buffer

(actual address will be *tbp0* x 64)**short** *tbw*,

Width of texture buffer

(actual width will be *tbw* x 64)**short** *psm*,

Format in which texture pixels are saved

0: PSMCT32

1: PSMCT24

2: PSMCT16

10: PSMCT16S

19: PSMT8

20: PSMT4

27: PSMT8H

36: PSMT4HL

44: PSMT4HH

48: PSMZ32

49: PSMZ24

50: PSMZ16

58: PSMZ16S

short *w*, *h*,

Width, height of texture

(Actual size will be 2^w and 2^h)**short** *tfx*

Texture functions

0 MODULATE

1 DECAL

2 HIGHLIGHT

3 HIGHLIGHT2

short *cbp*,

Base address of CLUT data

(actual address will be *cbp* x 64)**short** *cpsm*,

Format in which CLUT entries are saved

0: PSMCT32

1: PSMCT24

2: PSMCT16

10: PSMCT16S

short *cld*,

Loading of CLUT buffer

0: Do not load

1: Load from *cbp*2: Load from *cbp* and enter *cbp* value in the CBP0 register of the GS3: Load from *cbp* and enter *cbp* value in the CBP1

short filter)

register of the GS
 4: If CBP0!=*cbp*, load and set CBP0 to *cbp*
 5: If CBP1!=*cbp*, load and set CBP1 to *cbp*
 Filtering method
 0: NEAREST
 1: LINEAR
 2: NEAREST_MIPMAP_NEAREST
 3: NEAREST_MIPMAP_LINEAR
 4: LINEAR_MIPMAP_NEAREST
 5: LINEAR_MIPMAP_LINEAR

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads texture settings information for context 1 into the members of the `sceGsTexEnv` structure.

When a GIFtag (PACKED mode, REGS=A+D) is placed in memory immediately before the structure, information can be transferred directly to the GIF.

Note that if the structure is cached, data will not be transferred properly unless the D-cache is flushed back to memory.

When transferring a new texture to the GS, the texture should be reset using a `sceGsTexEnv` structure that was generated with flush set to 1.

Notes

This function loads the following values into the members of the `sceGsTexEnv` structure. Other values can be used if necessary.

```
tex01.TBP0 = tbp0
tex01.TBW = tbw
tex01.PSM = psm
tex01.TW = w
tex01.TH = h
tex01.TCC = 1
tex01.TFX = tfx
tex01.CBP = cbp
tex01.CPSM = cpsm
tex01.CLD = cld
tex11.MMAG = filter & 1
tex11.MMIN = filter
clamp1.WMS = 1
clamp1.WMT = 1
```

Return value

Returns the size of the `sceGsTexEnv` structure (in qwords).

sceGsSetDefTexEnv2

Generate texture environment settings

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

int sceGsSetDefTexEnv2(

sceGsTexEnv2 **tp*,

Address of texture information settings structure

short *flush*,

Disable texture page buffer

0: Do not disable

1: Disable

short *tbp0*,

Base address of texture buffer

(actual address will be *tbp0* x 64)

short *tbw*,

Width of texture buffer

(actual width will be *tbw* x 64)

short *psm*,

Format in which texture pixels are saved

0: PSMCT32

1: PSMCT24

2: PSMCT16

10: PSMCT16S

19: PSMT8

20: PSMT4

27: PSMT8H

36: PSMT4HL

44: PSMT4HH

48: PSMZ32

49: PSMZ24

50: PSMZ16

58: PSMZ16S

short *w*, *h*,

Width, height of texture

(Actual size will be 2^w and 2^h)

short *tfx*

Texture functions

0: MODULATE

1: DECAL

2: HILIGHT

3: HILIGHT2

short *cbp*,

Base address of CLUT data

(actual address will be *cbp* x 64)

short *cpsm*,

Format in which CLUT entries are saved

0: PSMCT32

1: PSMCT24

2: PSMCT16

10: PSMCT16S

short *cld*,

Loading of CLUT buffer

0: Do not load

1: Load from *cbp*2: Load from *cbp* and enter *cbp* value in the CBP0 register of the GS3: Load from *cbp* and enter *cbp* value in the CBP1 register of the GS4: If CBP0!=*cbp*, load and set CBP0 to *cbp*5: If CBP1!=*cbp*, load and set CBP1 to *cbp***short** *filter*)

Filtering method

0: NEAREST

1: LINEAR

2: NEAREST_MIPMAP_NEAREST

3: NEAREST_MIPMAP_LINEAR

4: LINEAR_MIPMAP_NEAREST

5: LINEAR_MIPMAP_LINEAR

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function loads texture settings information for context 2 into the members of the `sceGsTexEnv2` structure.

When a GIFtag (PACKED mode, REGS=A+D) is placed in memory immediately before the structure, information can be transferred directly to the GIF. Note that if the structure is cached, data will not be transferred properly unless the D-cache is flushed back to memory.

When transferring a new texture to the GS, the texture should be reset using a `sceGsTexEnv2` structure that was generated with flush set to 1.

Notes

This function loads the following values into the members of the `sceGsTexEnv2` structure. Other values can be used if necessary.

`tex02.TBP0 = tbp0``tex02.TBW = tbw``tex02.PSM = psm``tex02.TW = w``tex02.TH = h``tex02.TCC = 1``tex02.TFX = tfx``tex02.CBP = cbp``tex02.CPSM = cpsm``tex02.CLD = cld``tex12.MMAG = filter & 1``tex12.MMIN = filter``clamp2.WMS = 1`

clamp2.WMT = 1

Return value

Returns the size of the sceGsTexEnv2 structure (in qwords).

sceGsSetHalfOffset

Generate drawing offset data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax**void sceGsSetHalfOffset(****sceGsDrawEnv1** *draw,

Address of structure for setting up drawing environment

short centerx, **short** centery,

Coordinate at center of drawing area

short halfoff)

Offset addition control

0: Do not add

1: Add

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function calculates the drawing offset coordinate from the center coordinate of the drawing area and sets up xyoffset1 in the drawing environment structure.

If *halfoff* is set to 1, the offset is incremented by 8 in the y direction as a half-pixel increment.

Notes

In interlaced mode, the apparent vertical resolution can be made to be double the number of scan lines by drawing frames where the odd fields and the even fields are shifted by half a pixel.

If the buffer is set to FRAME mode, the images can be drawn shifted by a half-pixel by shifting the offset values for both fields by a half-pixel.

Return value

None

sceGsSetHalfOffset2

Generate drawing offset data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

```
void sceGsSetHalfOffset2(
    sceGsDrawEnv2 *draw,           Address of structure for setting up drawing environment
    short centerx, short centery,   Coordinate at center of drawing area
    short halfoff)                 Offset addition control
                                   0: Do not add
                                   1: Add
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function calculates the drawing offset coordinate from the center coordinate of the drawing area and sets up *xyoffset2* in the drawing environment structure for context 2.

If *halfoff* is set to 1, the offset is incremented by 8 in the y direction as a half-pixel increment.

Notes

In interlaced mode, the apparent vertical resolution can be made to be double the number of scan lines by drawing frames where the odd fields and the even fields are shifted by half a pixel.

If the buffer is set to FRAME mode, the images can be drawn shifted by a half-pixel by shifting the offset values for both fields by a half-pixel.

Return value

None

sceGsSwapDBuff

Swap double buffers

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 6, 2002

Syntax

int sceGsSwapDBuff(

sceGsDBuff *db,

Address of structure containing double buffer settings

int id)

Buffer number (only lowermost bit is valid)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Using the *id* argument, this function sets up the GS for one of the two drawing environments and display environments in the double buffer information structure.

This function can only be used to set the context 1 environment.

Notes

If the *db* double buffer information structure is cached, this function should be called after flushing the D-cache back to memory.

Return value

Returns the termination status.

When termination is normal, returns 0.

When there is a timeout (when a preceding Ch.2 DMA has not finished), returns -1.

sceGsSwapDBuffDc

Swap double buffers

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 6, 2002

Syntax

```
int sceGsSwapDBuffDc(
    sceGsDBuffDc *db,           Address of structure containing double buffer settings
    int id)                     Buffer number (only lowermost bit is valid)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Using the *id* argument, this function sets up the GS for one of the two drawing environments and display environments in the double buffer information structure.

This function can be used to set both the context 1 and context 2 environments.

Notes

If the *db* double buffer information structure is cached, this function should be called after flushing the D-cache back to memory.

Return value

Returns the termination status.

When termination is normal, returns 0.

When there is a timeout (when a preceding Ch.2 DMA has not finished), returns -1.

sceGsSyncPath

Wait for data transfer to finish

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	December 6, 2002

Syntax

```
int sceGsSyncPath(
```

int *mode*,

Wait mode

0: Block until wait condition is released

1: Don't block and return the busy status of the devices

u_short *timeout*)

Timeout counter

0: System default (approximately 4 VSynCs)

Positive value: Timeout interval (in HSyncs) (* not supported yet)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

If the *mode* argument is set to 0, the program will be blocked until the devices in the datapath (PATH1, 2, 3) are idle. If the transfer is not completed during the interval specified by *timeout*, a message will be sent to standard output and the function will exit with an error. If this happens, locked devices will remain locked and nothing will be reset.

The timeout argument specifies the waiting interval in maximum number of HSynCs. If timeout is specified as 0, the function will wait for a 4 VSync (1050 HSync) interval. If the *mode* argument is specified as 1, the busy status of the devices in the datapath at that instant will be returned.

With the *mode* argument set to 0, when the function completes normally, the contents of the registers are printed on the debug console.

D1_CHCR, D1_TADR, D1_MADR, D1_QWC

D2_CHCR, D2_TADR, D2_MADR, D2_QWC

VIF1_STAT, GIF_STAT

Return value

Returns the termination status.

When $mode == 0$,

When termination is normal, returns 0.

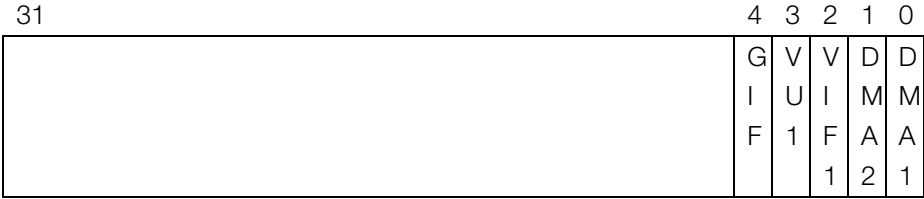
When termination is abnormal (a timeout occurs), returns a negative value.

When $mode == 1$,

Returns the busy status (positive value) of the following devices.

When the wait condition is released, returns 0.

Figure 2-1



0: idle
1: busy

sceGsSyncV

Wait for sync with V-Blank

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

```
int sceGsSyncV(
    int mode)                Reserved (specify 0)
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (must be called in an interrupt enabled state)

Description

Blocks program until a V-Blank interval starts.

The *mode* argument is provided for a future extension. For the current version, the argument should always be set to 0.

This function cannot be used together with the EE kernel service functions `AddIntcHandler(INTC_VBLANK_S, ,)` or `AddIntcHandler2(INTC_VBLANK_S, ,)`. Use this function together with `sceGsSyncVCallback` instead of `AddIntcHandler(INTC_VBLANK_S, ,)` or `AddIntcHandler2(INTC_VBLANK_S, ,)`.

Return value

Returns the FIELD information for interlaced mode.

Figure 2-2

Value	Result
0	Even field
1	Odd field (A 1 is always returned for non-interlaced mode)

sceGsSyncVCallback

Set up VSync callback

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libgraph	1.1	January 22, 2003

Syntax

```
int *sceGsSyncVCallback(
    int (*func)(int))          Entry address for callback function
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets up the callback function which is called at the start of a V-Blank interval. The previous setting will be discarded.

The entry address of the function will be saved as the sceGsVSCfunc global variable.

Notes

Calling sceGsResetGraph(0) will clear the Vsync callback function setting.

This function uses the AddIntcHandler kernel service function. If the callback function returns a -1, the other Vsync interrupt handlers registered using AddIntcHandler may not be called.

Since func functions are executed as interrupt handlers, special care is required when programming. Refer to the "Interrupt Handler Descriptions" section of \overview\eekernel for details.

The interrupt handler that was previously registered can be deleted by specifying NULL for the argument.

If a previously registered interrupt handler exists, and a second interrupt handler is registered, the function will internally delete the first interrupt handler and it will not be called. Use the kernel service function AddIntcHandler() to register second and subsequent interrupt handlers. DisableIntc() should be used to temporarily disable interrupts before adding an interrupt handler. Afterwards, use EnableIntc() to re-enable interrupts.

This function can be used together with sceGsSyncV.

Return value

Returns the entry address for the callback function that was previously set up.

Chapter 3: DMA Packet Management Services

Table of Contents

Functions	3-3
sceHiDMADel_Chain	3-3
sceHiDMADeleteMarked	3-4
sceHiDMAGet_BufferPtr	3-5
sceHiDMAGet_ChainAddr	3-6
sceHiDMAGetChainHead	3-7
sceHiDMAGetChainTail	3-8
sceHiDMAInit	3-9
sceHiDMAInit_DBuf	3-10
sceHiDMAMake_CallID	3-11
sceHiDMAMake_CallPtr	3-12
sceHiDMAMake_ChainEnd	3-13
sceHiDMAMake_ChainStart	3-14
sceHiDMAMake_ContinueMicro	3-15
sceHiDMAMake_DBufEnd	3-16
sceHiDMAMake_DBufStart	3-17
sceHiDMAMake_DynamicChainEnd	3-18
sceHiDMAMake_DynamicChainStart	3-19
sceHiDMAMake_ExecMicro	3-20
sceHiDMAMake_ExecMicroAddr	3-21
sceHiDMAMake_LoadGS	3-22
sceHiDMAMake_LoadGSLump	3-23
sceHiDMAMake_LoadImm	3-24
sceHiDMAMake_LoadMicro	3-25
sceHiDMAMake_LoadPtr	3-26
sceHiDMAMake_LoadStep	3-27
sceHiDMAMake_Lump	3-28
sceHiDMAMake_LumpEnd	3-29
sceHiDMAMake_LumpStart	3-30
sceHiDMAMake_WaitMicro	3-31
sceHiDMAMarkup	3-32
sceHiDMAPurge	3-33
sceHiDMARegist	3-34
sceHiDMARegistClear	3-35
sceHiDMASend	3-36
sceHiDMASendI	3-37
sceHiDMASet_BufferPtr	3-38
sceHiDMASwap	3-40
sceHiDMAWait	3-41

Functions

sceHiDMADel_Chain

Delete chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiDMADel_Chain(
 sceHiDMAChainID_t *id*) Chain ID to be deleted

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function allows buffers that were deleted and freed to be reused by the DMA Service.

No operation will be performed when a non-existent ID is specified.

Return value

Table 3-1

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiDMADeleteMarked

Delete marked chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax

void sceHiDMADeleteMarked(void)

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function completely deletes the chain that was previously marked using sceHiDMAMarkup().

Return Value

None

sceHiDMAGet_BufferPtr

Get packet buffer pointer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiDMAGet_BufferPtr(****u_int **addr)** Current buffer pointer value**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns in *addr* the address where the next sceHiDMAMake~ function will create the packet in the DMA Buffer.

addr will be set to NULL if this function is called without first calling sceHiDMAMake_ChainStart() or sceHiDMAMakeDynamicChainStart().

Return value

Table 3-2

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiDMAGet_ChainAddr

Get chain address

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiDMAGet_ChainAddr(
    sceHiDMAChainID_t id,           ID of chain for which address is to be obtained
    u_int **ptr)                   Obtained address storage location
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionThis function returns the starting address of the chain indicated by *id*.

Proper operation is not guaranteed if an invalid ID is passed as an argument.

Return value**Table 3-3**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiDMAGetChainHead

Get starting address of chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax

```
qword *sceHiDMAGetChainHead(
    sceHiDMAChainID_t id)          Chain ID for which address is to be obtained
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns the starting address of the DMA chain indicated by *id*.

When *id* == 0, this function returns the beginning of the Regist Chain.

Return Value

qword * Starting address of DMA chain

sceHiDMAGetChainTail

Get ending address of chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax

```
qword *sceHiDMAGetChainTail(
    sceHiDMAChainID_t id)           Chain ID for which address is to be obtained
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns the ending address of the DMA chain indicated by *id*. When *id* == 0, this function returns the beginning of the Regist Chain.

The meaning of "ending address" here is the empty address that is used when creating the next DMA Tag/Packet, not the address of the DMA Tag/Packet that was created last.

Although there won't be a problem if an arbitrary Tag (END Tag or RET Tag) is inserted in this return value address, note that only 1 qword is guaranteed to be inserted.

In addition, a sceHiDMAMake_-type packet creation function cannot be called after a tag is inserted by the user application.

Return Value

qword * Ending address of DMA chain

sceHiDMAInit

Initialize DMA

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiDMAInit(
    void (*func-alloc) (size_t, saize_t),           Function for allocating buffer memory
    void (*func-free)(void*),                     Function for freeing allocated memory
    size_t bsize)                                   Buffer byte size
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function performs HiG DMA Service initialization.

This function should be called only once by an application that uses the HiG DMA Service.

When *func* is set to NULL, sceHiMemAlign() and sceHiMemFree() are used.

Also, the function assigned by *func* must be equivalent to the function void * memalign(size_t BOUNDARY, size_t SIZE), which can allocate memory according to align and has for its arguments the number of align bytes for BOUNDARY and the allocation byte size for SIZE.

Example:

Specify 1K for a buffer. Use libc malloc.

```
sceHiDMAInit(memalign, free, 1024 * 1024);
```

Return value

Table 3-4

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	DMA buffer area allocation failed

sceHiDMAInit_DBuf

Initialize double buffer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr sceHiDMAInit_DBuf(

int start, Beginning of Double Buffer

```
int end)                                End of Double Buffer
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function makes settings for performing double buffering in VU1's VU MEM.

The *start* and *end* arguments indicate the start and end of the Double Buffer, respectively.

Note that these arguments are addresses that are specified in units of qwords. (That is, they must be values from 0 to 1024.)

Buffering should be selected by the micro code itself (XTOP instruction).

When setting data in the Double Buffer, addresses should be specified by considering the start of the Double Buffer as address 0.

Also, call `sceHiDMAMakeDBufStart()` before creating a data transfer chain in the Double Buffer, and call `sceHiDMAMakeDBufEnd()` after creating the data transfer chain.

All transfer instruction chains between StartDBuf and EndDBuf are created as transfer instructions in the Double Buffer.

Example:

Use the area from 120 to the end as the Double Buffer

```
sceHiDMAInIt DBuf(120, 1024);
```

This function creates a cnt VIF Tag, BASE, OFFSET, and VIF Command in the DMA Buffer.

Return value

Table 3-5

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_CallID

Create packet for calling chain with ID

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr sceHiDMAMake_CallID(
sceHiDMAChainID_t id) ID of chain to be called

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for calling a previously created chain.

This function creates two call Tags and NOP VIF Commands in the DMA Buffer.

Return value

Table 3-6

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	- sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called. - The relevant DMA Chain could not be found.

sceHiDMAMake_CallPtr

Create packet for calling a chain with a pointer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

```
sceHiErr sceHiDMAMake_CallPtr(
    u_int ptr)                Pointer to chain to be called
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for calling a previously created chain located at *ptr*.

This function creates two call VIF Tags and NOP VIF Commands in the DMA Buffer.

Return value

Table 3-7

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_ChainEnd

End chain creation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiDMAMake_ChainEnd(******id***)Address of variable that stores ID for referencing
created packet chain**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function declares the end of packet chain creation.

sceHiDMAMakeChainStart() and sceHiDMAMakeChainEnd() functions must be paired in a one-to-one correspondence.

If an attempt is made to end the creation of a packet chain without one having been started, an error will be returned.

The generated packet chain is later controlled using the *id*.**Return value****Table 3-8**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_ChainStart

Start chain creation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiDMAMake_ChainStart(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function declares the start of packet chain creation.

sceHiDMAMakeChainStart() and sceHiDMAMakeChainEnd() functions must be paired in a one-to-one correspondence.

If the creation of a packet chain has been started and not ended, an error will be returned.

Return value**Table 3-9**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was already called.

sceHiDMAMake_ContinueMicro

Restart execution of micro code

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr sceHiDMAMake_ContinueMicro(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates a packet for restarting a micro code program that has been stopped.

This function creates a cnt VIF TAG, NOP, MSCNT, and VIF Command in the DMA Buffer.

Return value

Table 3-10

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_DBufEnd

End creation of double buffer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiDMAMake_DBufEnd(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function declares the end of transfer instruction packet creation in the Double Buffer.

Return value**Table 3-11**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	sceHiDMAMake_DBufStart() was not called.

sceHiDMAMake_DBufStart

Start Double Buffer creation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiDMAMake_DBufStart(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function declares the start of transfer instruction packet creation in the Double Buffer.

To transfer a chain to the Double Buffer, call this function immediately before the step for creating that chain.

All packet creation instructions up until sceHiDMAMakeD_BufEnd() is executed are assumed to be instructions in the Double Buffer.

When transferring to the Double Buffer, addresses should be specified as relative addresses with the start of the Double Buffer assumed to be 0.

Return value**Table 3-12**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	sceHiDMAInit_DBuf() was not called.

sceHiDMAMake_DynamicChainEnd

End dynamic chain creation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiDMAMake_DynamicChainEnd(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function is used to declare the end of dynamic packet creation.

The sceHiDMAMake_DynamicChainStart() and sceHiDMAMake_DynamicChainEnd() functions must be paired in one-to-one correspondence.

If an attempt is made to end dynamic chain creation without it having been started, an error is returned.

Regist is automatically performed for the packet chain that was created and operations with id are not accepted. In addition, after the packet chain is transferred with sceHiDMASend(), the buffer that was used is automatically freed.

Return value**Table 3-13**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_DynamicChainStart

Start dynamic chain creation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiDMAMake_DynamicChainStart(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function is used to declare the start of dynamic packet creation.

The sceHiDMAMake_DynamicChainStart() and sceHiDMAMake_DynamicChainEnd() functions must be paired in one-to-one correspondence.

If an attempt is made to start dynamic chain creation without an end, an error is returned.

Return value

Table 3-14

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was already called.

sceHiDMAMake_ExecMicro

Start execution of micro code

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax**sceHiErr sceHiDMAMake_ExecMicro(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for executing the MICRO CODE that is in the Micro Memory of VU1.

The starting execution location is fixed at 0x0000.

This function creates a cnt VIF Tag, MSCAL, NOP, and VIF Command in the DMA Buffer.

Return value**Table 3-15**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_ExecMicroAddr

Start execution of microcode at specified address

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.5	December 6, 2002

Syntax

sceHiErr sceHiDMAMake_ExecMicroAddr(int addr) Vu1 MicroMem address of microcode to be executed

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for executing the MICRO CODE that is in the Micro Memory of VU1.

The starting execution location is passed in addr.

This function creates a cnt VIF Tag, MSCAL, NOP, and VIF Command in the DMA Buffer.

Return value**Table 3-16**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_LoadGS

Transfer data to the GS

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax**sceHiErr sceHiDMAMake_LoadGS(****u_int *ptr,**

EE transfer-source address

size_t qsize)

Transfer amount (qword size)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for directly transferring the amount of data specified by *qsize* starting from *ptr* of the EE, to the GS.

A suitable GIF tag must be attached to the data that is located at *ptr*.

This function creates a ref VIF Tag, NOP, DIRECT, and VIF Command in the DMA Buffer.

Return value**Table 3-17**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_LoadGSLump

Transfer data to GS

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	December 6, 2002

Syntax**sceHiErr sceHiDMAMake_LoadGSLump(****u_int *ptr,**

EE transfer-source address

size_t qsize)

Transfer amount (qword size)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Creates a command packet for directly transferring the data starting from the EE *ptr* address to the GS in sections of *qsize*.

This function is different from `sceHiDMAMake_LoadGS()` in that the data starting from *ptr* is not referenced directly but is used after copying it to the packet buffer in sections of *qsize*.

This function is used when you do not want to make a separate memory allocate, or when transferring data etc. that is created locally within the function.

This function creates a cnt VIF Tag, NOP, DIRECT, and VIF Command in the DMA Buffer.

Return value**Table 3-18**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_LoadImm

Transfer data with an immediate value

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax**sceHiErr sceHiDMAMake_LoadImm**

u_int *addr, VU1 transfer-destination address
qword imm) qword immediate value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for transferring the qword immediate value *imm* to the VU1 address specified by *addr*.

This function creates a cnt VIF Tag, STCYCL, UNPACK, and VIF Command in the DMA Buffer.

Return value**Table 3-19**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_LoadMicro

Transfer micro code

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax**sceHiErr sceHiDMAMake_LoadMicro(****char *code,**

Micro code address

size_t qsize)

Micro code qword size

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates a transfer instruction packet for loading a sequence of micro instructions into the MICRO MEMORY of VU1.

The loading location currently is fixed at micro memory address 0x0000.

This function creates a ref VIF Tag, MPG, NOP, and VIF Command in the DMA Buffer.

Return value**Table 3-20**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_LoadPtr

Transfer data with a pointer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax**sceHiErr sceHiDMAMake_LoadPtr(**

u_int * <i>addr</i> ,	VU1 transfer-destination address
u_int <i>ptr</i> ,	EE transfer-source address
size_t <i>qsize</i>)	Transfer volume (qword size)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for transferring the amount of data specified by *qsize* starting from *ptr* of the EE, to the VU1 address specified by *addr*.

This function creates a ref VIF Tag, STCYCL, UNPACK, and VIF Command in the DMA Buffer.

Return value**Table 3-21**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_LoadStep

Transfer data with offset

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr sceHiDMAMake_LoadStep(

u_int *addr,	VU1 transfer-destination address
u_int *ptr,	EE transfer-source address
size_t qsize,	Total size of transfer data (qword size)
int n,	Transfer-source partition size
int ofs)	Transfer-destination skip amount

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for loading an amount of data specified by *qsize*, starting from *ptr* of the EE to the VU1 address specified by *addr*, while skipping an interval of *ofs* every *n* qwords.

Example: Load 100 qwords starting at 0x00200000 of the EE to 0x00 of VU1 while skipping one qword every one qword.

```
sceHiLoadVU1_step(0x00, 0x00200000, 100, 1, 1)
```

This function creates a ref VIF Tag, STCYCL, UNPACK, and VIF Command in the DMA Buffer.

Return value

Table 3-22

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMake_Lump

Create packet with immediate data string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr **sceHiDMAMake_Lump**(
qword *imm*) Immediate data value (qword)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

See `sceHiDMAMake_LumpStart()`.

This function copies the immediate value that was assigned to *imm* into the DMA Buffer.

Return value

Table 3-23

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_LumpStart() was not called.

sceHiDMAMake_LumpEnd

End packet creation with immediate data string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr sceHiDMAMake_LumpEnd(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

See sceHiDMAMake_LumpStart().

This function creates a cnt VIF Tag, STCYCL, UNPACK, and VIF Command in the DMA Buffer.

Return value

Table 3-24

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	sceHiDMAMake_LumpStart() was not called.

sceHiDMAMake_LumpStart

Start packet creation with immediate data string

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiDMAMake_LumpStart(
    u_int *addr)                VU1 transfer-destination address
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet for transferring a data string in one lump. The data string should have been assigned by a `sceHiDMAMake_Lump()` function, which was called between the `sceHiDMAMake_LumpStart()` and `sceHiDMAMake_LumpEnd()` functions. These functions are used when sending consecutive immediate data values to the VU1 location `addr`, for which the start of lump transfer was declared (Lump transfer start declaration).

LumpStart and LumpEnd cannot be nested.

Return value

Table 3-25

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_LumpStart() was already called.

sceHiDMAMake_WaitMicro

Wait for micro code

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax**sceHiErr sceHiDMAMake_WaitMicro(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an instruction packet that waits for the micro code program to stop.

This function creates a cnt VIF Tag, FLUSH, NOP, and VIF Command in the DMA Buffer.

Return value**Table 3-26**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
4	SCE_HIG_FAILURE	sceHiDMAMake_ChainStart() or sceHiDMAMake_DynamicChainStart() was not called.

sceHiDMAMarkup

Mark a chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax

```
void sceHiDMAMarkup(
    sceHiDMAChainType_t type)          Chain type to be marked
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function marks all previously created chains of the type specified by *type*.

Marking a chain enables operations to be performed on those chains later.

The three types that can be specified are Regist Chain, Static Chain, and Dynamic Chain.

Return Value

None

sceHiDMAPurge

Reconfigure chain buffer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax**sceHiErr sceHiDMAPurge(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function exists only for compatibility with earlier versions. Currently, there is no need to call it.

Return value

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiDMARegister

Register chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr sceHiDMARegister(
sceHiDMAChainID_t id) ID of chain for which a transfer is to be registered

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function registers a chain that was created during the packet chain creation step as an actual transfer chain.

The contents of every frame transfer must be registered.

This function creates two call VIF Tags and NOP VIF Commands in the DMA Buffer.

Return value

Table 3-27

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow
2	SCE_HIG_INVALID_VALUE	Invalid argument

sceHiDMARegistClear

Delete DMA transfer registration

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax**void sceHiDMARegistClear(void)****Calling Conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function cancels all registrations that were performed with sceHiDMARegist().

It can be used after a registration is performed when a scene has changed and you do not want to use the registered drawing or when you want the user application to perform transfer processing (see HiG Service Overview).

Return Value

None

sceHiDMASend

DMA transfer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	December 6, 2002

Syntax

sceHiErr sceHiDMASend (void);

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function actually performs DMA transfers of a transfer registration chain that was created by repeatedly executing sceHiDMARegist(). This function also flushes the cache internally.

To perform transfers within an interrupt handler, use the sceHiDMASendI() function.

To wait for the end of transfer, either use sceHiDMAWait() or a function that performs an equivalent operation.

```
Example << Entire Flow >>
foo_init()
{
    sceHiDMAInit(...);
    sceHiDMAMake_ChainStart();
    << making packet ...
    sceHiDMAMake_ChainEnd(&global_id1);
    sceHiDMAMake_ChainStart();
    << making packet ...
    sceHiDMAMake_ChainEnd(&global_id2);
}
main()
{
    foo_init()
    while (1) {
        sceHiDMARegist(global_id1);
        sceHiDMARegist(global_id2);
        sceHiDMASend();
        sceHiDMAWait();
    }
}
```

Return value

Table 3-28

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow

sceHiDMASendI

Perform DMA transfer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	December 6, 2002

Syntax**sceHiErr sceHiDMASendI(void)****Calling conditions**

Can be called from an interrupt handler

Cannot be called from a thread

Description

This function enables sceHiDMASend() to be called from within an interrupt handler.

Return value

Table 3-29

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	DMA buffer overflow

sceHiDMASet_BufferPtr

Set buffer pointer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiDMASet_BufferPtr(
    u_int *addr)                BufferPointer
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the Buffer Pointer that is used by the HiG DMA Service.

Normally, the buffer pointer is managed automatically. However, this function is required to modify a previously created packet.

This function is used together with sceHiDMAGetPtr().

The user is responsible for returning the Buffer Pointer to its original value.

Example:

After lumping is specified for data[0], data[1], data[2], and data[3], change the contents of data[2].

```
int foo (void)
{
    .
    .
    .
    sceHiDMAMake_LumpStart(lump_addr);
    sceHiDMAMake_Lump(data[0]);
    sceHiDMAMake_Lump(data[1]);
    sceHiDMAGetPtr(change_addr);
    sceHiDMAMake_Lump(data[2]);
    sceHiDMAMake_Lump(data[3]);
    sceHiDMAMake_LumpEnd();
    .
    .
    .
    if (is_change_2) {
        sceHiDMAGetPtr(save_ptr);
        sceHiDMASetPtr(change_addr);
        sceHiDMAMake_Lump(change_data);
        sceHiDMASetPtr(save_ptr);
    }
    .
    .
    .
}
```


Return value**Table 3-30**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiDMASwap

Swap DMA registration buffer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiDMASwap(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function swaps the buffer used for chains for which transfer was reserved using the sceHiDMARegister() function. It enables DMA chains to be created and registered in the background of DMA transfer processing.

Return value**Table 3-31**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiDMAWait

Wait for end of DMA transfer

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiDMAWait(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function waits for the end of DMA transfer processing. Its action is equivalent to that of `sceGsSyncPath(0, 0)`.

When using the DMA service to perform DMA transfers, always call `sceHiDMAWait()` to wait for the transfer to end.

Return value**Table 3-32**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Timeout

Chapter 4: High Level Graphics Library

Table of Contents

Structures	4-3
sceHiData	4-3
sceHiErr	4-4
sceHiErrStateType	4-5
sceHiHeadData	4-6
sceHiHeader	4-7
sceHiList	4-8
sceHiPlug	4-9
sceHiPlugTable	4-10
sceHiType	4-11
Functions	4-12
sceHiAddDataBlk	4-12
sceHiAddPlugBlk	4-13
sceHiCalcPlugBlkSize	4-14
sceHiCallPlug	4-15
sceHiContPlugListStatus	4-16
sceHiContPlugStatus	4-17
sceHiCreateDataBlk	4-18
sceHiCreatePlugBlk	4-19
sceHiGetData	4-20
sceHiGetDataPlace	4-21
sceHiGetInsPlug	4-22
sceHiGetList	4-23
sceHiGetPlug	4-24
sceHiGetPlugList	4-25
sceHiGetPlugPlace	4-26
sceHiGetType	4-27
sceHiInsDataBlk	4-28
sceHiInsPlugBlk	4-29
sceHiMakeDataBlk	4-30
sceHiMakeType	4-31
sceHiNewPlugBlk	4-32
sceHiParseHeader	4-33
sceHiRegistTable	4-34
sceHiRmvDataBlk	4-35
sceHiRmvPlugBlk	4-36
sceHiSetDataType	4-37
sceHiSetPluginApi	4-38
sceHiSetPlugType	4-39
sceHiStopPlugListStatus	4-40
sceHiStopPlugStatus	4-41

Structures

sceHiData

Data block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	July 2, 2001

Structure

```
typedef struct _sceHiData {  
    sceHiType type;  
    char count;  
  
    char reserve[3];  
    u_int size;  
    u_int data[1];  
} sceHiData;
```

Data type of the data

Number of plug-ins that are currently using this data (part of system management information)

Reserved area for future expansion

Data size (byte size)

Data (variable-length array)

Description

This is the data block type.

sceHiErr

Error

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 26, 2001

Structure

```
typedef enum _sceHiErr
    SCE_HIG_NO_ERR,
    SCE_HIG_NO_HEAP,
    SCE_HIG_INVALID_VALUE,
    SCE_HIG_INVALID_DATA,
    SCE_HIG_FAILURE,
} sceHiErr;
```

Description

The sceHiErr type is the ERROR type that is returned by library functions and plug-in functions.

The library sets values from 0 to 127 as library-reserved ERRORS.

Values of 128 and higher are handled as ERROR numbers specific to various plug-ins.

ERRORs having values of 128 and higher should be judged according to sceHiErrState information and ERROR number.

The following values can be entered for this variable.

Table 4-1

Constant	Meaning
SCE_HIG_NO_ERR	Normal (no ERROR)
SCE_HIG_NO_HEAP	Insufficient heap size
SCE_HIG_INVALID_VALUE	Invalid value
SCE_HIG_INVALID_DATA	Invalid data
SCE_HIG_FAILURE	Processing failure

sceHiErrStateType

Error status

Library	Introduced	Documentation last modified
libhig	2.1	March 26, 2001

Structure

```
typedef struct _sceHiErrStateType {
    sceHiPlug *top;           Top parent plug-in of plug-in where ERROR occurred
    sceHiPlug *plug;         Plug-in where ERROR occurred
    int process;              Process number where ERROR occurred
    sceHiType type;           Type attribute of plug-in where ERROR occurred
    const char *mes;          ERROR message
} sceHiErrStateType;
```

Description

This is a structure for returning information about an ERROR that occurred within a plug-in.

The following variable exists as a global variable.

```
sceHiErrStateType    sceHiErrState;
```

When an error is returned while a function that returns the sceHiErr type is being used, the state can be determined from this global variable.

For error details, refer to the *mes* contents.

sceHiHeadData

Header data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	January 4, 2001

Structure

```
typedef struct _hig_head_data_t {
    char plug_name[12];           Plug-in block-specific name
    struct _sceHiPlug *plug_blk_addr;  Pointer to plug-in block
} sceHiHeadData;
```

Description

This is the plug-in registration portion of the data format header information.
It indicates the type of the plug-in block to be used.

sceHiHeader

Header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	January 4, 2001

Structure

```
typedef struct _hig_head_t {
    u_int ver;                libhig version
    u_int reserve1;           Reserved area for future expansion
    u_int reserve2;           Reserved area for future expansion
    u_int qsize;              Header size (qword size)
} sceHiHeader;
```

Description

First qword of data format header
This is managed by the library.

sceHiList

Block list

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	July 2, 2001

Structure

```
typedef struct _sceHiList {
    sceHiType type;           Identification code
    u_int *addr;              "Relative/absolute" pointer to inserted plug-in
                              block/data block
    u_int reserve;            Reserved area for future expansion
} sceHiList;
```

Description

This is a structure for enumerating the inserted plug-in blocks and data blocks.
addr is initially a relative address. Parsing is performed to rewrite it as an absolute address.

sceHiPlug

Plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	July 2, 2001

Structure

```
typedef struct _sceHiPlug {
    sceHiType type;
    void *myapi;
    u_int size;
    char nplug;

    char ndata;

    char reserve[6];
    u_int stack;

    u_int args;

    qword list;
} sceHiPlug;
```

Description

This is the plug-in block type.

For a virtual plug-in, the myapi member will contain NULL.

The args member acts as an interface for exchanging information between the user and plug-in function.

Arguments are passed to the plug-in by entering values in the args member and exchanging the information.

sceHiPlugTable

Plug-in table

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	January 4, 2001

Structure

```
typedef struct _sceHiPlugTable {
    sceHiType type;           Type attribute of plug-in
    void *func;               Pointer to plug-in function
} sceHiPlugTable;
```

Description

This is a plug-in registration structure that becomes an argument of sceHiRegistTable().

The type information of a plug-in is associated in a one-to-one fashion with a function pointer. Plug-ins which are not required should be deleted from this table.

sceHiType

Type information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	July 2, 2001

Structure

```
typedef struct _sceHiType {
    unsigned long repository:8;      Plug-in and data repository identifier (constant value)
    unsigned long project:8;         Plug-in and data project identifier (constant value)
    unsigned long category:8;        Plug-in and data category identifier (constant value)
    unsigned long status:8;          Plug-in and data status information (bit packed)
    unsigned long id:24;              Plug-in and data identifier
    unsigned long revision:8;         Plug-in and data revision number
} sceHiType;
```

Description

An application programmer specifies a plug-in and data for the library by creating this sceHiType structure and passing it as an argument.

Functions

sceHiAddDataBlk

Add data block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiAddDataBlk(

```
sceHiPlug *plug;
```

Destination plug-in block

```
sceHiData *data;)
```

Source data block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function adds a data block to a plug-in block by finding an empty list location.

If no empty location is found, an error is returned.

Return value

Table 4-2

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	No empty location in data block list

sceHiAddPlugBlk

Add plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiAddPlugBlk(****sceHiPlug** *plug1; Destination plug-in block**sceHiPlug** *plug2;) Source data block**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function adds the source plug-in block to the destination plug-in block by finding an empty list location.

If no empty location is found, an error is returned.

Return value**Table 4-3**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	No empty location in plugin block list

sceHiCalcPlugBlkSize

Calculate plug-in block size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax

sceHiErr sceHiCalcPlugBlkSize(

int <i>nplug</i> ,	Number of plug-ins
int <i>ndata</i> ,	Number of data values
size_t * <i>size</i>)	Area for returning calculation result (in bytes)

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns the size in bytes of a plug-in block having a plug-in list of *nplug* plug-ins and a data list of *ndata* data values. It is used when allocating an area which will require a plug-in, such as `sceHiCreatePlugBlk()`.

Return Value

Table 4-4

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiContPlugListStatus

Resume plug-in function activation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiContPlugListStatus(**sceHiList** **list*)

Plug-in block list

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function resumes the activation of plug-in functions kept by the plug-in block at the beginning of the plug-in block list.

The fourth bit of the type attribute status is 0.

This function is called when `sceHiCallPlug()` is called.

To inhibit function activation, call `sceHiStopPlugListStatus()`.

Return value

Table 4-6

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiContPlugStatus

Resume plug-in function activation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiErr sceHiContPlugStatus(
    sceHiPlug *plug)          Plug-in block
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function resumes the activation of plug-in functions kept by the plug-in block.

The fourth bit of the type attribute status is 0.

This function is called when sceHiCallPlug() is called.

To inhibit function activation, call sceHiStopPlugStatus().

Return value

Table 4-7

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiCreateDataBlk

Create data block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax

sceHiErr sceHiCreateDataBlk(

sceHiData <i>*data</i> ,	Actual data that is not in a data block
const u_int <i>*rdata</i> ,	Generated data block
sceHiType <i>*type</i>)	Type attribute to be set

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function generates a data block from actual data that is not in a data block and the specified type attribute, in a manner similar to `sceHiMakeDataBlk()`.

It differs from `sceHiMakeDataBlk()` in that memory is not allocated internally by the function. Therefore, the caller must allocate and free the area. The size of the area to be allocated is `sizeof(sceHiData)`.

The status tag part of the *type* field will become `is_ref_data_bit=1`.

Return Value

Table 4-8

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiCreatePlugBlk

Create plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.7	December 6, 2002

Syntax**sceHiErr sceHiCreatePlugBlk(**

sceHiPlug *plug,	Plug-in block area to be created
int nplug,	Maximum number of plug-ins to be registered
int ndata,	Maximum number of data values to be registered
const sceHiType *type)	Plug-in type to be registered

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an empty plug-in block, in a manner similar to sceHiNewPlugBlk().

It differs from sceHiNewPlugBlk() in that memory is not allocated internally by the function. Therefore, the caller must allocate and free the area. Use sceHiCalcPlugBlkSize() to determine the size of the area to be allocated.

Return Value

Table 4-9

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGetData

Get data block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiGetData(

sceHiPlug <i>*plug;</i>	Plug-in block address
u_int <i>**data;</i>	Address of variable for receiving RAW data address
sceHiType <i>type;</i>)	Type attribute that is the object of the search

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function searches the data blocks that are held by the plug-in block for the type attribute that matches the type argument value. It returns in the data argument the address of the RAW data that is maintained by the data block.

Return value

Table 4-10

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Specified type and data type do not match
4	SCE_HIG_FAILURE	Could not find specified type of data block

sceHiGetDataPlace

Get data block list number

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiGetDataPlace(

sceHiPlug **plug*, Address of plug-in block to check
sceHiData **data*, Address of data block to check
int **ofs*) List number of data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns in *ofs* the list number position at which the address specified by *data* exists in the data list at the address specified by *plug*.

The *ofs* value that is returned is a list number that can be handled by a function such as `sceHiInsDataBlk()`.

If the address specified by *data* does not exist in the data list at the address specified by *plug*, an error is returned.

Return value

Table 4-11

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

However, when it cannot be found, *ofs* is set to -1.

sceHiGetInsPlug

Get inserted plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGetInsPlug(

sceHiPlug **plug*,

Original plug-in block

```
sceHiPlug **plug2,
```

Inserted plug-in block to be obtained

sceHiType *type*)

Inserted plug-in type attribute to be obtained

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns the address of the plug-in block that matches the specified type attribute among the inserted plug-in blocks.

Return value

Table 4-12

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Specified type and plug-in type do not match
4	SCE_HIG_FAILURE	Could not find specified type of plug-in block

sceHiGetList

Get block list

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiGetList(

sceHiPlug * <i>plug</i> ;	Plug-in block address
sceHiList ** <i>list</i> ;	Address of variable for receiving the list item address
sceHiType <i>type</i>);	Type attribute that is the object of the search

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function stores in the list argument the address of a list item whose type attribute matches the value specified by the type argument, among the list items of the inserted plug-in block list and data block list maintained by the plug-in block.

Return value

Table 4-13

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Could not find specified type of data block

sceHiGetPlug

Get plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiGetPlug(  
    u_int *data;           Data format header address  
    char *name;            Plug-in block-specific name  
    sceHiPlug **plug;)    Address of variable for receiving the plug-in block  
                           address
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function stores in plug the address of the plug-in block having the specified name, from among the plug-ins that are registered in the data format header at the address specified by *data*.

The plug-in block is needed in order to start up a plug-in function using sceHiCallPlug(), which is described later.

Return value

Table 4-14

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Could not find plug-in block

sceHiGetPlugList

Get plug-in block list

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGetInsPlug(

sceHiPlug *plug, Original plug-in block
sceHiList **list, Plug-in block list to be obtained
sceHiType type) Plug-in type attribute to be obtained

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns the address of the plug-in block list that matches the specified type attribute among the inserted plug-in blocks.

Return value

Table 4-15

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Could not find specified type of plug-in block list

sceHiGetPlugPlace

Get plug-in block list number

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiGetPlugPlace(

sceHiPlug <i>*plug1,</i>	Address of plug-in block to check
sceHiPlug <i>*plug2,</i>	Address of plug-in block to check
int <i>*ofs)</i>	List number of plug2

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns in *ofs* the list number position at which the plug-in block specified by *plug2* exists in the plug-in list of the plug-in block specified by *plug1*.

The *ofs* value that is returned is a list number that can be handled by a function such as `sceHilnsPlugBlk()`.

If the plug-in block specified by *plug2* does not exist in the plug-in list specified by *plug1*, an error is returned.

Return value

Table 4-16

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

However, if it cannot be found, *ofs* is set to -1.

sceHilnsDataBlk

Insert data block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiInsDataBlk(
```

sceHiPlug <i>*plug;</i>	Destination plug-in block
--------------------------------	---------------------------

```
sceHiData *data;
```

Source data block

```
int nb;) List number
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function inserts the data block at the specified list number location in the plug-in block.

If the specified list number location is not empty, an error is returned.

Return value

Table 4-18

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Same type of data block list already exists

sceHilnsPlugBlk

Insert plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```

sceHiErr sceHilnsPlugBlk(
    sceHiPlug *plug1;           Destination plug-in block
    sceHiPlug *plug2;           Source plug-in block
    int nb;)                     List number

```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function inserts the specified plug-in block at the specified list number location in the plug-in block.

If the specified list number location is not empty, an error is returned.

Return value

Table 4-19

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Same type of plug-in block list already exists

sceHiMakeDataBlk

Create data block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr **sceHiMakeDataBlk**(
 u_int **rdata*; Real data having no data block type
 sceHiData ***arg*; Data block that is created
 sceHiType **type*;) Type attribute to be set

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates a data block from the specified type attribute and real data that doesn't have a data block type.

The status tag part of the *type* field will be *is_ref_data_bit*=1.

Memory is allocated internally from the HIG heap. To destroy it, use *sceHiMemFree*()).

Return value

Table 4-20

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Failed to allocate area for new data block

sceHiMakeType

Generate type information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiMakeType(
    sceHiType *src_type;           Type structure to be packed
    u_long *pack_type;)           Address of variable that will contain packed data
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function packs sceHiType type data to convert it to a form that will be used internally.

For information about packed bit images, refer to the type information for the high level graphics library data format.

Currently, when the sceHiType member is a bit field, this function has no meaning. Therefore, be sure to call this function only when it is specifically required.

Return value

Table 4-21

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_NO_HEAP	Failed to allocate memory internally

sceHiNewPlugBlk

Create plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiNewPlugBlk(

int <i>nplug</i> ;	Maximum number of plug-ins to be registered
int <i>ndata</i> ;	Maximum number of sets of data to be registered
sceHiPlug ** <i>plug</i> ;	Address of empty plug-in block that will be created
sceHiType * <i>type</i>)	Type of plug-in to be registered

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates an empty plug-in block
Memory is allocated internally from the HIG heap.
To destroy it, use sceHiMemFree().

Return value

Table 4-22

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_NO_HEAP	Failed to allocate memory area internally

sceHiParseHeader

Analyze data format

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
sceHiErr sceHiParseHeader(
    u_int *data;)           Data format header address
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function analyzes the data format starting at the header address.

Relative addresses within the data are converted to real addresses.

After this function is called, internal plug-in blocks and data blocks become available for use.

Data for which this function has not been called cannot be used directly.

Conversion from the header address to a plug-in block is done recursively.

Return value

Table 4-23

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	Version does not match
4	SCE_HIG_FAILURE	- Number of entries in plug-in block list is invalid - Number of entries in data block list is invalid

sceHiRegistTable

Register plug-in function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiRegistTable(
 sceHiPlugTable *table; Plug-in registration table
 u_int num); Number of tables

Calling conditions

- Cannot be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function uses the plug-in registration table to register a plug-in function.
A plug-in function that has not been registered cannot be used.

Return value

Table 4-24

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiRmvDataBlk

Remove data block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiRmvDataBlk(**

sceHiPlug **plug*;
 Plug-in block from which data block registration is to be deleted

sceHiData **data*;)
 Target data block for which registration is to be deleted

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function deletes the list information of the specified data block from the plug-in block.

The specified data block itself is not deleted.

The deleted list location is assumed to be empty, and a data block can be added or inserted at that location.

Return value

Table 4-25

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Nothing exists to be deleted

sceHiRmvPlugBlk

Remove plug-in block

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiRmvPlugBlk(

```
sceHiPlug *plug1;
```

Plug-in block from which plug-in block registration is to be deleted

```
sceHiPlug *plug2;)
```

Target plug-in block for which registration is to be deleted

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function deletes the list information of the specified plug-in block from the plug-in block.

The specified plug-in block itself is not deleted.

The deleted list location is assumed to be empty, and a plug-in block can be added or inserted at that location.

Return value

Table 4-26

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Nothing exists to be deleted

sceHiSetPluginApi

Set plug-in function

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiSetPluginApi(

```
sceHiPlug *plug;)
```

Plug-in block for which plug-in function is to be set

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function registers a plug-in function for a plug-in block.

The plug-in function will be the function that was specified with the *type* attribute.

Return value

Table 4-28

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiSetPlugType

Set plug-in block type attribute

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax**sceHiErr sceHiSetPlugType(****sceHiPlug** *plug; Plug-in block for which type attribute is to be set**sceHiType** *htype;) Type attribute to be set**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionThis function sets the *type* attribute for the plug-in block that was created with sceHiNewPlugBlk().**Return value**

Table 4-29

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiStopPlugListStatus

Inhibit plug-in function activation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiStopPlugListStatus(

sceHiList **list*) Plug-in block list

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function inhibits the activation of plug-in functions kept by the plug-in block at the beginning of the plug-in block list.

The fourth bit of the type attribute status is 1.

This function will not be called even if `sceHiCallPlug()` is called.

To resume function activation, call `sceHiContPlugListStatus()`.

Return value

Table 4-30

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiStopPlugStatus

Inhibit plug-in function activation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiErr sceHiStopPlugStatus(
    sceHiPlug *plug)           Plug-in block
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function inhibits the activation of plug-in functions kept by the plug-in block.

The fourth bit of the type attribute status is 1.

This function will not be called even if sceHiCallPlug() is called.

To resume function activation, call sceHiContPlugStatus().

Return value

Table 4-31

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

Chapter 5: GS Register Management Services

Table of Contents

Common Structures	5-5
sceHiGsGiftag	5-5
Display Environment Setting Structures	5-6
sceHiGsDisplay	5-6
GS Local Memory Management Structures	5-7
sceHiGsMemTbl	5-7
Context Management Structures	5-8
sceHiGsCtx	5-8
sceHiGsEnv	5-12
Old Structures	5-13
sceHiGsContext	5-13
sceHiGsGeneral	5-14
sceHiGsPacked	5-15
sceHiGsPacked_t	5-16
Display Environment Setting Functions	5-17
sceHiGsDisplayEnd	5-17
sceHiGsDisplayMode	5-18
sceHiGsDisplaySet	5-19
sceHiGsDisplaySize	5-20
sceHiGsDisplayStatus	5-21
sceHiGsDisplaySwap	5-22
GS Local Memory Management Functions	5-23
sceHiGsBlockSize	5-23
sceHiGsMemAddTbl	5-24
sceHiGsMemAlloc	5-25
sceHiGsMemFree	5-26
sceHiGsMemInit	5-27
sceHiGsMemPrintTbl	5-28
sceHiGsMemRealloc	5-29
sceHiGsMemRestSize	5-30
sceHiGsMemRestSizePlus	5-31
sceHiGsPageSize	5-32
GS Register Setting Functions	5-33
sceHiGsCtxChkSize	5-33
sceHiGsCtxCopy	5-34
sceHiGsCtxCreate	5-35
sceHiGsCtxDelete	5-37
sceHiGsCtxFcache	5-38
sceHiGsCtxFcacheI	5-39
sceHiGsCtxGetDefault	5-40
sceHiGsCtxGetRect	5-41
sceHiGsCtxGetTex0	5-42
sceHiGsCtxRegist	5-43
sceHiGsCtxRegistClear	5-44

sceHiGsCtxSend	5-45
sceHiGsCtxSendClear	5-46
sceHiGsCtxSetByDBuff	5-47
sceHiGsCtxSetClearColor	5-48
sceHiGsCtxSetClearMode	5-49
sceHiGsCtxSetClearZ	5-50
sceHiGsCtxSetColorDepth	5-51
sceHiGsCtxSetContext	5-52
sceHiGsCtxSetDefault	5-53
sceHiGsCtxSetDefaultValues	5-54
sceHiGsCtxSetDepth	5-55
sceHiGsCtxSetEachBuffer	5-57
sceHiGsCtxSetFrame	5-58
sceHiGsCtxSetHalfOffset	5-59
sceHiGsCtxSetLumpBuffer	5-60
sceHiGsCtxSetMax	5-61
sceHiGsCtxSetRect	5-62
sceHiGsCtxSetRegAlpha	5-64
sceHiGsCtxSetRegClamp	5-65
sceHiGsCtxSetRegFba	5-66
sceHiGsCtxSetRegMiptbp1	5-67
sceHiGsCtxSetRegMiptbp2	5-68
sceHiGsCtxSetRegTest	5-69
sceHiGsCtxSetRegTex0	5-70
sceHiGsCtxSetRegTex1	5-71
sceHiGsCtxSetRegXyoffset	5-72
sceHiGsCtxSetRegZbuf	5-73
sceHiGsCtxSetZbufDepth	5-74
sceHiGsCtxShiftBuffers	5-76
sceHiGsCtxSwap	5-77
sceHiGsCtxSwapAll	5-78
sceHiGsCtxUpdate	5-79
sceHiGsEnvCopy	5-80
sceHiGsEnvCreate	5-81
sceHiGsEnvDelete	5-83
sceHiGsEnvFcache	5-84
sceHiGsEnvFcache1	5-85
sceHiGsEnvGetDefault	5-86
sceHiGsEnvRegist	5-87
sceHiGsEnvSend	5-88
sceHiGsEnvSetDefault	5-89
sceHiGsEnvSetRegBitblt	5-90
sceHiGsEnvSetRegColclamp	5-91
sceHiGsEnvSetRegDimx	5-92
sceHiGsEnvSetRegDthe	5-93
sceHiGsEnvSetRegFog	5-94
sceHiGsEnvSetRegFogcol	5-95
sceHiGsEnvSetRegLabel	5-96
sceHiGsEnvSetRegPabe	5-97
sceHiGsEnvSetRegPrmode	5-98
sceHiGsEnvSetRegPrmodecont	5-99

sceHiGsEnvSetRegScanmsk	5-100
sceHiGsEnvSetRegSignal	5-101
sceHiGsEnvSetRegTexa	5-102
sceHiGsEnvSetRegTexclut	5-103
sceHiGsEnvSetRegTrxdir	5-104
sceHiGsEnvSetRegTrxpos	5-105
sceHiGsEnvSetRegTrxreg	5-106
sceHiGsEnvUpdate	5-107
sceHiGsServiceExit	5-108
sceHiGsServiceInit	5-109
sceHiGsServiceSetRegistFunc	5-110
Old GS Register Setting Functions	5-110
sceHiGsAlphaRegs	5-110
sceHiGsClampRegs	5-112
sceHiGsClear	5-113
sceHiGsClearColor	5-114
sceHiGsClearDepth	5-115
sceHiGsColclampRegs	5-116
sceHiGsContextID	5-117
sceHiGsContextStatus	5-118
sceHiGsDimxRegs	5-119
sceHiGsDtheRegs	5-120
sceHiGsFbaRegs	5-121
sceHiGsFogcolRegs	5-122
sceHiGsFogRegs	5-123
sceHiGsFrameRegs	5-124
sceHiGsGeneralStatus	5-125
sceHiGsMiptbp1Regs	5-126
sceHiGsMiptbp2Regs	5-127
sceHiGsPabeRegs	5-128
sceHiGsPackedCreate	5-129
sceHiGsPackedDelete	5-130
sceHiGsPackedUpdate	5-131
sceHiGsPrmodecontRegs	5-132
sceHiGsPrmodeRegs	5-133
sceHiGsTestRegs	5-134
sceHiGsTex0Regs	5-135
sceHiGsTex1Regs	5-136
sceHiGsTexaRegs	5-137
sceHiGsXyoffsetRegs	5-138
sceHiGsZbufRegs	5-139

Common Structures

sceHiGsGiftag

Giftag structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 26, 2001

Structure

```
typedef struct _sceHiGsGiftag{
    unsigned long nloop:15;           giftag nloop field
    unsigned long eop:1;              giftag eop field
    unsigned long id:30;               DMA packet ID
    unsigned long pre:1;               giftag pre field
    unsigned long prim:11;             giftag prim field
    unsigned long flg:2;               giftag flag field
    unsigned long nreg:4;              giftag nreg field
    unsigned long regs:64;             giftag regs field
} sceHiGsGiftag;
```

Description

This is the GIFTag management structure.

The ID from the DMA packet management function is placed in the *id* member.

Display Environment Setting Structures

sceHiGsDisplay

Display structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 26, 2001

Structure

```
typedef struct _sceHiGsDisplay{  
    int swap;                               Swap value for the double buffer  
    sceGsDBuff dbuf;                        Structure maintaining double buffer information  
                                           For details, refer to libgraph.  
} sceHiGsDisplay;
```

Description

This structure maintains information used for switching between the drawing and display environments. It is used by the GS display function. For calling libgraph internally, it contains the sceGsDBuff structure as a member.

GS Local Memory Management Structures

sceHiGsMemTbl

GS memory table structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 26, 2001

Structure

```
typedef struct _sceHiGsMemTbl{
    u_int align;
    u_int addr;
    u_int size;
    struct _sceHiGsMemTbl *next;
} sceHiGsMemTbl;
```

Alignment size (in words)
Either of the following:
SCE_HIGS_PAGE_ALIGN = 2048
SCE_HIGS_BLOCK = 64

Address of the reserved region (in words)

Size of the reserved region (in words)

Pointer to the next GS memory table
Constitutes the chunk table

Description

This is a structure for use in GS local memory management used by GS memory management functions.

Reserves size sections starting from *addr* with that alignment.

The chunk table is constructed from the *next* member.

The next member is NULL for the ending table.

Context Management Structures

sceHiGsCtx

Management structure for handling context register setting group

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	July 2, 2001

Structure

```
typedef struct {
    sceHiGsClearPacket clearp;           Clear setting/transfer area
    sceHiGsPacked packed;               Register transfer area
    sceHiGsContext value;               Register setting area
    u_short fbp[2];                     Frame buffer pointer
    u_short validregs;                  Transfer register
    u_char clearmode;                   Clear mode
    u_char ctxt;                        Used context
    u_char swap;                        Current buffer when double buffer is used
    u_char field;                       Current field when interlace is used
    u_char isDbuf;                      Double buffer flag (cannot be changed)
    u_char isSync;                      Synchronous flag
    u_char isInterlace;                 Interlace flag
    u_char isZbuf;                      Z-buffer flag
    char ppos[2];                       Internal management area (cannot be changed)
} sceHiGsCtx
```

Description

This is a management structure for handling the context register setting group among the GS register management services.

It is acquired with the `sceHiGsCtxCreate()` function and freed with the `sceHiGsCtxDelete()` function.

The timing for setting and using each member is described below.

clearp (clear setting/transfer area)

The `clearp` area is set by the `sceHiGsCtxSetClear*()` and `sceHiGsCtxCopy()` functions.

In addition, the `sceHiGsCtxUpdate()` function copies the value of `value->test` to a portion of `clearp->clear.testb`.

If the `Send` or `Regist` functions are called with the `clear` argument set to 1 when the clear mode is other than `SCE_HIGS_CLEAR_KEEP`, the `clearp` area will either be transferred to the GS or a transfer will be registered.

The `clearp` area is transferred after the `packed` area.

packed (register transfer area)

The packed area is allocated and initialized by the `sceHiGsCtxCreate()` function.

The contents of the packet, which have been reflected in the settings of other members, are updated by the `sceHiGsCtxUpdate()` function. In addition, some of the settings are updated directly by the `Swap` function.

The packed area is not copied by the `sceHiGsCtxCopy()` function.

If the `Send` or `Regist` functions are called, the packed area will either be transferred to the GS or a transfer will be registered.

The packed area is transferred before the *clearp* area.

value (Register setting area)

The value area is set by the `Set` and `sceHiGsCtxCopy()` functions. The contents of the member may be changed directly, however, `value.frame.FBP` is not used. The other contents are reflected in the transfer area by the `sceHiGsCtxUpdate()` function for the registers specified by *validregs*.

fbp[2] (Frame buffer pointer)

The `fbp` member is set by the `sceHiGsCtxSetFrame()` and `sceHiGsCtxCopy()` functions. It is used only when the `SCE_HIGS_VALID_FRAME` bit is set ON in *validregs*.

The `fbp[swap]` value is reflected in the transfer area of the *packed* member by the `sceHiGsCtxUpdate()` function in accordance with the value of the *swap* member. It is also reflected in the transfer area of the packed member by the `sceHiGsCtxSwap()` function in accordance with the value of the *swap* argument assigned to the function when the double buffer flag is 1, as well as by the `sceHiGsCtxSwapAll()` function when the synchronous flag is also 1.

validregs (Transfer register)

The bits of the *validregs* member specify the registers to be transferred.

The transfer setting bits of related registers are automatically set ON in the `Set` function. The *validregs* member is also copied by the `sceHiGsCtxCopy()` function. Since no function is currently provided for the user to specify these registers, the following variables should be used.

To transfer the frame register:	<code>SCE_HIGS_VALID_FRAME</code>
To transfer the zbuf register:	<code>SCE_HIGS_VALID_ZBUF</code>
To transfer the tex0 register:	<code>SCE_HIGS_VALID_TEX0</code>
To transfer the tex1 register:	<code>SCE_HIGS_VALID_TEX1</code>
To transfer the tex2 register:	<code>SCE_HIGS_VALID_TEX2</code>
To transfer the mip1 register:	<code>SCE_HIGS_VALID_MIPTBP1</code>
To transfer the mip2 register:	<code>SCE_HIGS_VALID_MIPTBP2</code>
To transfer the clamp register:	<code>SCE_HIGS_VALID_CLAMP</code>
To transfer the test register:	<code>SCE_HIGS_VALID_TEST</code>
To transfer the alpha register:	<code>SCE_HIGS_VALID_ALPHA</code>
To transfer the xyoffset register:	<code>SCE_HIGS_VALID_XYOFFSET</code>
To transfer the scissor register:	<code>SCE_HIGS_VALID_SCISSOR</code>
To transfer the fba register:	<code>SCE_HIGS_VALID_FBA</code>

Example: To transfer only the frame register and zbuf register:

```
sceHiGsCtx *gsctx;
```

```
gsctx->validregs = SCE_HIGS_VALID_FRAME | SCE_HIGS_VALID_ZBUF;
```

Example: To not transfer the clamp register:

```
sceHiGsCtx *gsctx;
gsctx->validregs &= ~(u_short)SCE_HIGS_VALID_CLAMP;
```

clearmode (Clear mode)

The clearmode member keeps the value that was set by the sceHiGsSetClearMode() and sceHiGsCtxCopy() functions.

It is used to determine whether the clear mode is SCE_HIGS_CLEAR_KEEP in the sceHiGsCtxUpdate(), Send, and Regist functions.

ctxt (Used context)

The ctxt member is set by the sceHiGsCtxSetContext() and sceHiGsCtxCopy() functions.

It is reflected in the context of the registers that are used by the sceHiGsCtxUpdate() function.

swap (Current buffer for double buffering)

The swap member is set by the sceHiGsCtxSwap() function when the double buffer flag is 1 and by the sceHiGsCtxSwapAll() function when the synchronous flag is also 1. It is also copied by the sceHiGsCtxCopy() function. It is used to select the frame buffer pointer by the Swap and sceHiGsCtxUpdate() functions when the double buffer flag is 1.

When the double buffer flag is 0 (when a single buffer is used), the swap member must be set to zero.

field (Current field for interlace)

The field member is set by the sceHiGsCtxSwap() function when the interlace flag is 1 and by the sceHiGsCtxSwapAll() function when the synchronous flag is also 1. It is also copied by the sceHiGsCtxCopy() function.

This member is used to determine whether the current field is TOP or BOTTOM by the Swap and sceHiGsCtxUpdate() functions.

isDbuf (Double buffer flag)

The isDbuf member is set only by the sceHiGsCtxCreate() function. It cannot be changed any other way, and is not copied by the sceHiGsCtxCopy() function. It is used to determine whether or not a double buffer is in use.

isSync (Synchronous flag)

The isSync member is temporarily set to 1 by the sceHiGsCtxCreate() function when double buffering is specified and set to 0 otherwise.

Its value is copied by the sceHiGsCtxCopy() function.

Since no function is currently provided for changing this value, you must set the member variable directly to 0 or 1 when necessary.

This member is used by the sceHiGsCtxSwapAll() function to automatically determine whether or not to call the Swap function.

isInterlace (Interlace flag)

Normally, the isInterlace member is set directly to 0 or 1. Its value is also copied by the sceHiGsCtxCopy() function.

The value is temporarily set automatically in the sceHiGsCtxSetByDbuff() function according to the current interlace mode. The interlace mode, which is set by the second argument of the sceResetGraph() function, is also set by the sceHiGsDisplayMode() function within the HiG library.

This value is used to determine whether or not an interlace offset should be reflected in the YOFFSET value in the sceHiGsCtxUpdate() and sceHiGsCtxSwap() functions.

isZbuf (Z-buffer flag)

The isZbuf member is set according to the argument specification in the sceHiGsCtxSetDepth(), sceHiGsCtxSetZbufDepth(), and sceHiGsCtxSetDefaultValues functions.

The value is also copied by the sceHiGsCtxCopy() function.

It is used by the sceHiGsCtxChkSize() function to calculate the GS memory size that the context will use.

char ppos[2] (Internal management area (cannot be modified))

This is an internal management area that cannot be modified.

sceHiGsEnv

GS register management service / environment group management structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	July 2, 2001

Structure

```
typedef struct {
    sceHiGsPacked packed;           Register transfer area
    u_long *value;                   Register setting area
    u_int validregs;                 Transfer register
} sceHiGsEnv
```

Description

This is a management structure for handling the environment register setting group among the GS register management services.

It is acquired with the `sceHiGsEnvCreate()` function and freed with the `sceHiGsEnvDelete()` function.

The timing for setting and using each member is described below.

packed (register transfer area)

The packed area is allocated and initialized in the `sceHiGsEnvCreate()` function.

The contents of the packet, which have been reflected in the settings of other members, are updated by the `sceHiGsEnvUpdate()` function.

This area is not copied by the `sceHiGsEnvCopy()` function. When the Send or Regist functions are called, the packed area will either be transferred to the GS or a transfer will be registered.

value (Register setting area)

The value area is allocated and initialized in the `sceHiGsEnvCreate()` function. It is set by the Set and `sceHiGsEnvCopy()` functions. The value set here is reflected in the transfer area by the `sceHiGsCtxUpdate()` function.

validregs (Transfer register)

The validregs member is specified by the argument of the `sceHiGsEnvCreate()` function. It cannot be modified by any other method.

The bits of the validregs member specify the registers to be transferred. It is used by the Set and `sceHiGsEnvUpdate()` functions, and is also used by the `sceHiGsEnvCopy()` function to determine whether or not copying can be performed.

Old Structures

sceHiGsContext

GS context register structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 26, 2001

Structure

```
typedef struct _sceHiGsContext{
    sceGsFrame frame;           FRAME register
    sceGsZbuf zbuf;            ZBUF register
    sceGsTex0 tex0;            TEX0 register
    sceGsTex1 tex1;            TEX1 register
    sceGsTex2 tex2;            TEX2 register
    sceGsMiptbp1 miptbp1;      MIPTBP1 register
    sceGsMiptbp2 miptbp2;      MIPTBP2 register
    sceGsClamp clamp;          CLAMP register
    sceGsTest test;            TEST register
    sceGsAlpha alpha;          ALPHA register
    sceGsXyoffset xyoffset;     XYOFFSET register
    sceGsScissor scissor;       SCISSOR register
    sceGsFba fba;              FBA register
} sceHiGsContext;
```

Description

This is a structure used for managing the group of general-purpose registers with 2 contexts.

The context can be switched between 1 and 2 by calling the function `sceHiGsContextID()`.

sceHiGsGeneral

GS general-purpose register structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 26, 2001

Structure

```
typedef struct _sceHiGsGeneral{
    sceGsColclamp colclamp;          COLCLAMP register
    sceGsDimx dimx;                  DIMX register
    sceGsDthe dthe;                  DTHE register
    sceGsFog fog;                    FOG register
    sceGsFogcol fogcol;              FOGCOL register
    sceGsPabe pabe;                  PABE register
    sceGsTexa texa;
    sceGsPrmode prmode;              PRMODE register
    sceGsPrmodecont prmodecont;      PRMODECONT register
    sceHiGsContext *context;         Current context register
} sceHiGsGeneral;
```

Description

This is the GS general-purpose register structure.

A pointer to the current context register structure is placed in the context member.

Switching is performed with `sceHiGsContextID()`.

sceHiGsPacked

Structure used for PACKED mode A+D format transfers

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 26, 2001

Structure

```
typedef struct sceHiGsPacked{
    sceHiGsGiftag *giftag;           GIFtag
    sceHiGsPacked_t *packed;         PACKED data
} sceHiGsPacked;
```

Description

Structure used for PATH2 transfers.

sceHiGsPacked_t

PACKED mode A+D format structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 26, 2001

Structure

```
typedef struct sceHiGsPacked_t{
    u_long data; /* :64 */           Output data
    u_char addr; /* :8 */           Output destination register
    unsigned long padd:56;          Padding
} sceHiGsPacked_t;
```

Description

Packing format A+D structure used in PACKED mode.

Display Environment Setting Functions

sceHiGsDisplayEnd

End display

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsDisplayEnd(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Frees GS memory reserved by sceHiGsDisplaySet.

Internally calls sceHiGsMemFree.

Return value

Table 5-1

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	Failed to free GS memory

sceHiGsDisplayMode

Set display mode

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsDisplayMode(

u_int mode)

Display mode specification

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Sets the value obtained from the OR of the enumerated types `sceHiGsReset_t`, `sceHiGsDisp_t`, `sceHiGsRGBA_t`, `sceHiGsDEPTH_t` in *mode*.

Internally calls the `sceGsResetGraph` function.

Passes the value of sceHiGsRGBA_t and sceHiGsDEPTH_t to sceHiGsDisplaySize described below.

Return value

Table 5-2

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	Mode specification incorrect

sceHiGsDisplaySet

Set display

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsDisplaySet(

u_int <i>w</i> ,	Width
u_int <i>h</i> ,	Height
u_int <i>psm</i> ,	Frame buffer storage format
u_int <i>zpsm</i>)	Depth buffer storage format

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Sets the double frame buffer and the double depth buffer using their respective storage formats.

Internally calls the sceGsSetDBuff function.

Internally calls the sceHiGsMemAlloc function.

Returns an error if reserving of the GS memory area fails.

Return value

Table 5-3

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Insufficient GS memory heap area
2	SCE_HIG_INVALID_VALUE	Failed to free GS memory

sceHiGsDisplaySize

Set display size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsDisplaySize(

u_int <i>width</i>	Width
u_int <i>height</i>)	Height

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Calls the `sceHiGsDisplaySet` function with the specified size by taking the `sceHiGsRGBA_t`, `sceHiGsDEPTH_t` enumerated members from `sceHiGsDisplayMode` described above.

If the values of `sceHiGsRGBA_t`, `sceHiGsDEPTH_t` are incorrect, an error is returned.

Return value

Table 5-4

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Insufficient GS memory heap area or failed to allocate internal memory area
2	SCE_HIG_INVALID_VALUE	Argument was incorrect or failed to free GS memory
4	SCE_HIG_FAILURE	Called after sceHiGsServiceInit() was called

sceHiGsDisplayStatus

Display structure state

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiGsDisplay *sceHiGsDisplayStatus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

Returns a pointer to the current display structure.

Return value

sceHiGsDisplay* Pointer to the display structure

sceHiGsDisplaySwap

Swap display buffers

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```
sceHiErr sceHiGsDisplaySwap(
    int field)
```

Next field specification

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Performs buffer swapping of the drawing area and the display area.

The specification of whether the next frame is even or odd at the time of interlace, is specified in the *field* argument.

Ignored in the case of non-interlace.

Calls the sceGsSetHalfOffset,sceGsSwapDBuff functions internally.

Return value

Table 5-5

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsMemAddTbl

Add GS memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsMemAddTbl(****sceHiGsMemTbl *tbl)**

Gs memory table to be added

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

The sceHiGsMemTbl structure tbl is added to the chunk table built using the sceHiGsMemAlloc function, etc.

tbl must be set with a correct value before calling this function.

The same area or an overlapping area can be added to the chunk table.

Return value

Table 5-6

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	tbl is NULL or invalid

sceHiGsMemFree

Free GS memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsMemFree(****sceHiGsMemTbl *tbl)**

Gs memory table to be freed

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionFrees the area of the argument *tbl*.

The structure of the chunk table is corrected.

Return value

Table 5-8

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	Argument is NULL or invalid

sceHiGsMemInit

Initialize GS memory management

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsMemInit(**

u_int *addr*, Starting address of the buffer being reserved (in words)

size_t *size*) Buffer size being reserved (in words)

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Specifies the buffer area of GS local memory being managed by the library.

Default is the entire area and results in *addr*=0x0, *size*=0x100000.**Return value**

Table 5-9

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Overflow of the specified area
2	SCE_HIG_INVALID_VALUE	<i>size</i> is 0

sceHiGsMemPrintTbl

Display GS memory area information

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsMemPrintTbl(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

Displays information for all tables for which areas have been reserved.

All the values are displayed in hexadecimal notation.

Return value

Table 5-10

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsMemRealloc

Reallocate GS memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```

sceHiGsMemTbl *sceHiGsMemRealloc(
sceHiGsMemTbl *tbl,           Gs memory table to be reallocated
u_int align,                  Area alignment (in words)
size_t size)                  Area size (in words)

```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Performs a reallocation of the area.

The *tbl* argument is freed temporarily after which the area is reserved.

In case of a failure, *tbl* remains in the freed state.

Return value

sceHiGsMemTbl* Returns a pointer to the table that allocated the GS memory area.

If there is an error, the following is returned

Table 5-11

Value	Result
NULL	GS memory area allocation failed - Failed to allocate internal memory area - Overflow in GS memory

sceHiGsMemRestSize

Residual size of GS memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**size_t sceHiGsMemRestSize(void)****Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

Returns the remaining size starting from the last address of the reserved area.

Return value

size_t Returns the remaining size (in words)

sceHiGsMemRestSizePlus

Available space in GS memory area

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

size_t sceHiGsMemRestSizePlus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

Returns the amount of empty space available in reserved areas up to the present time. This can be used to determine how much waste there is in reserved areas.

Return value

size_t Returns the available space (in words)

sceHiGsPageSize

Acquire page size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

size_t sceHiGsPageSize(

u_int <i>w</i> ,	Width
u_int <i>h</i> ,	Height
u_int <i>psm</i>)	Pixel storage format

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

Returns the size as a function of the page alignment (2Kword) in words.

Return value

Returns the size (in words)

GS Register Setting Functions

sceHiGsCtxChkSize

Calculate amount of GS memory used

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

u_int sceHiGsCtxChkSize(

sceHiGsCtx *gsctx)

Pointer to management structure of port for which calculation is to be performed

Calling conditions

None

Description

This function returns the required amount of GS memory to be used by the specified port. It is used when allocating GS memory and setting the buffer pointer. The pixel depth, the presence or absence of a Z-buffer and its depth, and screen size must be correctly set for the calculation.

Notes

This function performs its calculation assuming that the frame buffer and Z-buffer reside in GS memory in units of pages.

Return value

Returns the GS memory usage size (units: words)

sceHiGsCtxCopy

Copy port (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxCopy(****sceHiGsCtx** *dst, Pointer to management structure of destination port**sceHiGsCtx** *src) Pointer to management structure of source port**Calling conditions**

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function makes a copy of the contents of the port. When the source port has a double buffer setting, the source fbp[0] member is copied to the fbp[1] member. When the destination port has a single buffer setting, the value of the swap member will always be zero.

Since the transfer area is not copied, after this function is called, the transfer area must be updated by calling sceHiGsEnvUpdate() before the area is transferred.

Return value

Table 5-12

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Argument is invalid

sceHiGsCtxCreate

Create port (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiGsCtx *sceHiGsCtxCreate(
    int isDbuf)                Double buffer flag
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function acquires and initializes one context register group management structure and returns a pointer to it.

Notes

Initial values that are set within this function are as follows.

validregs (transfer register) 0 (no transfer register)

value (register setting area) frame.FBMSK: 0 (no mask, update)

alpha: (Cs-Cd)(O)As+Cd:

alpha.A=SCE_GS_ALPHA_CS

alpha.B=SCE_GS_ALPHA_CD

alpha.C=SCE_GS_ALPHA_AS

alpha.D=SCE_GS_ALPHA_CD

alpha.FIX=128

tex1: No mipmap, bilinear

tex1.LCM=0

tex1.MXL=0

tex1.MMAG=SCE_GS_LINEAR

tex1.MMIN=SCE_GS_LINEAR

tex1.L=0

tex1.K=0

test:

test.ATE no initial value

test.ATST=SCE_GS_ALPHA_ALWAYS

test.AREF=128

test.AFAIL=0

```

test.DATE=0
test.DATM=0
clamp:
clamp.WMS=SCE_GS_CLAMP
clamp.WMT=SCE_GS_CLAMP
clamp.MINU=0
clamp.MAXU=0
clamp.MINV=0
clamp.MAXV=0

fba.FBA=0 (no alpha correction)
clearmode (clear mode) SCE_HIGS_CLEAR_ALL
(clear all frame buffers and Z-buffers)

ctxt (context used)                0 (context 1)
swap (current buffer when double buffer is used) 0
field (current field when interlace is used)      0
isDbuf (double buffer flag)                    According to value of isDbuf argument
isSync (synchronous flag)                      1 (synchronize)
isInterlace (interlace flag)                   According to value of isDbuf argument

```

Return value

When processing is successful, a pointer to the management structure of the created context register group is returned.

When there is an error, the following is returned:

Table 5-13

Value	Result
NULL	Internal memory area allocation failed or an attempt was made to open more than the restricted number of ports.

sceHiGsCtxFcache

Flush transfer area cache (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxFcache(****sceHiGsCtx** *gsctxPointer to management structure of port for which
cache is to be flushed**int** clear)

0: Do not flush clear transfer area

1: Flush clear transfer area

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function flushes the transfer area cache of the specified port. It is called when sceHiGsCtxSend() is used immediately after the transfer area was updated by sceHiGsCtxUpdate() or sceHiGsCtxSwap().

Notes

sceHiGsCtxFcache() should be used within an interrupt handler.

Return value**Table 5-15**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxFcacheI

Flush transfer area cache (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxFcacheI (

sceHiGsCtx *gsctx,

Pointer to management structure of port for which
cache is to be flushed

int clear)

0: Do not flush clear transfer area

1: Flush clear transfer area

Calling conditions

Can be called from an interrupt handler

Cannot be called from a thread

Description

This function flushes the transfer area cache of the specified port. This function is called when sceHiGsCtxSend() is used for a DMA transfer immediately after the transfer area was updated by sceHiGsCtxUpdate() or sceHiGsCtxSwap().

Notes

This function is for use within the sceHiGsCtxFcache() interrupt handler.

Return value

Table 5-16

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxGetDefault

Get default port (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiGsCtx *sceHiGsCtxGetDefault(void)

Calling conditions

None

Description

This function gets the default port that was set by the sceHiGsCtxSetDefault() function.

Return value

Returns a pointer to the management structure of the context register group's default port.

sceHiGsCtxGetRect

Get screen size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxGetRect(

sceHiGsCtx *gsctx,

Pointer to management structure of relevant port

int *xyzw)

Memory address for which screen size is to be obtained.

Calling conditions

None

Description

This function gets the port's screen size. Since the screen size is calculated based on the scissor register, if the scissoring setting is a different size than the screen, the correct value will not be returned. The memory address (4 words) must be allocated in advance by the caller.

Notes

The contents that are set are as follows.

xyzw[0] = gsctx->value.scissor.SCAX0;

xyzw[1] = gsctx->value.scissor.SCAY0;

xyzw[2] = 1 + gsctx->value.scissor.SCAX1 - gsctx->value.scissor.SCAX0;

xyzw[3] = 1 + gsctx->value.scissor.SCAY1 - gsctx->value.scissor.SCAY0;

Return value

Table 5-17

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxGetTex0

Get tex0 register value for using port as texture

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxGetTex0(

sceHiGsCtx *gsctx,

Port for drawing texture

u_long *tex0,

Pointer indicating area for saving obtained register value

int swap,

When a double buffer is used, buffer ID of side getting register value.

When a single buffer is used, zero must be specified.

int tcc,

Texture component setting

int tfx)

Texture function setting

Calling conditions

None

Description

This function gets the tex0 register value when the port is to be directly used as a texture.

The actual data is output to the memory area indicated by tex0.

The destination memory (u_long size) must be allocated in advance by the caller.

If the port's screen size, buffer width, or frame buffer pointer have not been set correctly, an incorrect setting value will be returned.

Currently, a setting that uses a port having an offset from the buffer pointer (a port for which the upper left corner does not match the buffer pointer) cannot be used.

Return value

Table 5-18

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxRegist

Register transfer (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxRegist(

sceHiGsCtx *gsctx,

Pointer to management structure of port for which transfer is to be registered

int clear)

0: Do not register the clear transfer area

1: Register the clear transfer area

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function uses the transfer registration function to register a transfer of the transfer area for the specified port.

The transfer registration function is set with sceHiGsServiceSetRegistFunc().

By default, since the transfer is registered as an HiG DMA library dynamic chain, the actual transfer will occur when sceHiDMASend() is called later.

Even if *clear* is set to 1, a transfer need not be registered for the clear transfer area when the clear mode is SCE_HIGS_CLEAR_KEEP.

Notes

Whether or not multithreading can be used depends on the transfer registration function. Since in the initial state the registered function is not multithread safe, this function is also not multithread safe. However, if the registered function was multithread safe, this function will be multithread safe.

Return value

Table 5-19

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Transfer register function returned a non-zero value (error)

sceHiGsCtxRegistClear

Register transfer for clear transfer area only (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax

```
sceHiErr sceHiGsCtxRegistClear(
    sceHiGsCtx *gsctx)           Pointer to management structure of port for which
                                transfer is to be registered
```

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function uses the transfer registration function to register a transfer of the clear transfer area within the specified port.

The transfer registration function is set using `sceHiGsServiceSetRegistFunc()`.

By default, since the transfer is registered as an HiG DMA library dynamic chain, the actual transfer will occur when `sceHiDMASend()` is called later.

When the clear mode is `SCE_HIGS_CLEAR_KEEP`, a transfer cannot be registered.

Notes

Whether or not multithreading can be used depends on the transfer registration function. In the initial state, the registered function is not multithread safe, so this function will not be safe either. However, it will be safe when a multithread-safe function is registered.

Return value

Table 5-20

Value	Macro	Result
0	<code>SCE_HIG_NO_ERR</code>	Processing was successful
4	<code>SCE_HIG_FAILURE</code>	Transfer register function returned a non-zero value (error)

sceHiGsCtxSend

Transfer immediately (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSend(

sceHiGsCtx *gsctx,

Pointer to management structure of port for which transfer area is to be transferred

int clear)

0: Do not transfer clear transfer area

1: Transfer clear transfer area

Calling conditions

None

Description

This function uses the DMA ch.2 normal mode for GIF transfers to immediately transfer the transfer area of the specified port. Before calling this function, you must confirm that the transfer area cache is flushed and that DMA ch.2 is available. This function completes without confirming the end of the last DMA transfer.

Even if *clear* is set to 1, a transfer need not be registered for the clear transfer area when the clear mode is SCE_HIGS_CLEAR_KEEP.

Notes

When *clear* is set to 1 and the clear mode is other than SCE_HIGS_CLEAR_KEEP, the clear transfer area will be locked and the DMA transfer of this area will not begin until the DMA transfer of the register transfer area ends. To prevent this from happening, either transfer the clear transfer area independently using a separate function such as sceHiGsCtxSendClear(), or use the sceHiGsCtxRegister() function.

Return value

Table 5-21

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSendClear

Transfer clear transfer area immediately (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSendClear (

sceHiGsCtx *gsctx)

Pointer to management structure of port for which
clear transfer area is to be transferred

Calling conditions

None

Description

This function uses the normal mode of DMA ch.2 to immediately transfer the transfer area of the specified port to the GIF. Before calling this function, you must confirm that the transfer area cache is flushed and that DMA ch.2 is empty. This function ends without confirming that the last DMA transfer has completed.

When the clear mode is SCE_HIGS_CLEAR_KEEP, a transfer cannot be performed.

Return value

Table 5-22

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetClearColor

Set clear color

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetClearColor(**

sceHiGsCtx *gsctx,	Pointer to management structure of port for which value is to be set
u_char red,	R value
u_char green,	G value
u_char blue,	B value
u_char alpha)	A value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the sprite color to be used for clearing.

The setting contents are as follows.

```
gsctx->clearp.clear.rgbaq.R = red;
gsctx->clearp.clear.rgbaq.G = green;
gsctx->clearp.clear.rgbaq.B = blue;
gsctx->clearp.clear.rgbaq.A = alpha;
gsctx->clearp.clear.rgbaq.Q = 1.0f;
```

Return value**Table 5-24**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetClearMode

Set clear mode

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSetClearMode(

sceHiGsCtx *gsctx,,

Pointer to management structure of port for which value is to be set

u_int mode)

Clear mode specification

SCE_HIGS_CLEAR_KEEP

Do not clear

SCE_HIGS_CLEAR_COLOR

Clear only frame buffer

SCE_HIGS_CLEAR_DEPTH

Clear only Z-buffer

SCE_HIGS_CLEAR_RGB

Clear only RGB value within frame buffer (do not clear Alpha value)

SCE_HIGS_CLEAR_ALL

Clear both frame buffer and Z-buffer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the mode used when clearing.

When a value other than SCE_HIGS_CLEAR_KEEP is set for mode, the TEST register transfer setting will be valid. In this case, you must use a function such as sceHiGsCtxSetRegTest() separately to set a value for the TEST register.

Return value

Table 5-25

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetClearZ

Set clear depth

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetClearZ(****sceHiGsCtx** *gsctx,,Pointer to management structure of port for which
value is to be set**u_int** z,)

Z-value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the Z-value of the sprite to be used for clearing.

Return value

Table 5-26

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetColorDepth

Set frame buffer format

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiErr sceHiGsCtxSetColorDepth(
    sceHiGsCtx *gsctx,           Pointer to management structure of port for which
                                frame buffer format is to be set
    int psm)                     Frame buffer pixel format
```

Calling conditions

None

Description

This function sets the frame buffer format.

Example: For a 32-bit frame buffer

```
sceHiGsCtxSetColorDepth(gsctx, SCE_GS_PSMCT32);
```

Notes

The setting contents are as follows. Since a transfer is registered for the frame register, either set missing settings separately in this register or cancel the transfer registration.

```
gsctx->value.frame.PSM=psm;
gsctx->validregsl=SCE_HIGS_VALID_FRAME;
```

Values that are not set here in the frame register

frame: FBP, FBW, FBMSK (the initial value is also set for FBMSK when FBMSK is created)

Return value

Table 5-27

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetContext

Switch context

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSetContext(

sceHiGsCtx *gsctx,,

Pointer to management structure of port for which value is to be set

int id)

Context ID

0 Context 1

1 Context 2

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the context that the port sets. The default value is id=0 (context 1).

The value set here is used to determine the register context to be set later by the sceHiGsCtxUpdate() function.

Context 1 when id=0, and context 2 when id=1

Operation is not guaranteed when a value other than 0 or 1 is specified for id.

Return value

Table 5-28

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetDefault

Set default port (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSetDefault(

sceHiGsCtx *gsctx)

Pointer to management structure to be specified for default port

Calling conditions

None

Description

This function sets *gsctx* in the context register group's default port.

The default port set here is transferred to the GS within the *sceHiGsSwapDisplay()* function.

If *gsctx* is set to a NULL pointer, the standard port will be set.

Notes

The following processing is performed within the *sceHiGsSwapDisplay()* function for the default port that is set here.

```
sceHiGsCtx *port;
sceHiGsCtxSwapAll(swap, field); (When the synchronous flag is set, swap processing is performed)
port = sceHiGsCtxGetDefault();
sceHiGsCtxFcache(port, 1);      (Default port's transfer area cache is flushed)
sceHiGsCtxSend(port, 1);        (Default port is transferred)
```

Return value

Table 5-29

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetDefaultValues

Set screen size and buffer format

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiErr sceHiGsCtxSetDefaultValues(
    sceHiGsCtx *gsctx,           Pointer to management structure of port for which
                                values are to be set
    int psm,                     Frame buffer pixel format
    int zpsm,                     Z-buffer format
    int isZbuf,                   1: Use Z-buffer
                                0: Do not use Z-buffer
    int w,                       Screen width (units: pixels)
    int h)                       Screen height (units: pixels)
```

Calling conditions

None

Description

This function sets the screen size and buffer format. It enables the transfer of all transfer registers other than tex0, tex2, mip1, and mip2.

Notes

The setting contents are as follows.

```
gsctx->validregs |= SCE_HIGS_VALID_FRAME | SCE_HIGS_VALID_ZBUF |
```

```
    SCE_HIGS_VALID_TEX1 | SCE_HIGS_VALID_CLAMP |
    SCE_HIGS_VALID_TEST | SCE_HIGS_VALID_ALPHA |
    SCE_HIGS_VALID_XYOFFSET | SCE_HIGS_VALID_SCISSOR |
    SCE_HIGS_VALID_FBA;
```

```
sceHiGsCtxSetDepth(gsctx, psm, zpsm, isZbuf);
sceHiGsCtxSetRect(gsctx, 0, 0, w, h, SCE_HIGS_FBW_DEFAULT);
sceHiGsCtxSetLumpBuffer(gsctx, 0);
```

Return value

Table 5-30

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetDepth

Set buffer format

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiErr sceHiGsCtxSetDepth(
    sceHiGsCtx *gsctx,           Pointer to management structure of part for which
                                value is to be set
    int psm,                     Frame buffer pixel format
    int zpsm,                     Z-buffer format
    int isZbuf)                  1: Use Z-buffer
                                0: Do not use Z-buffer
```

Calling conditions

None

Description

This function sets the frame buffer and Z-buffer formats.

When the Z-buffer will not be used, a dummy value should be set for *zpsm*.

Example 1: When 32-bit frame buffer and 24-bit Z-buffer are to be used

```
sceHiGsCtxSetDepth(gsctx, SCE_GS_PSMCT32, SCE_GS_PSMZ24, 1);
```

Example 2: When 16-bit frame buffer and no Z-buffer are to be used

```
sceHiGsCtxSetDepth(gsctx, SCE_GS_PSMCT16, 0, 0);
```

Notes

The setting contents are as follows. Since transfers are registered for the frame, zbuf, and test registers, either set missing settings separately within these registers or cancel the transfer registration.

```
gsctx->isZbuf=isZbuf;
gsctx->value.frame.PSM=psm;
gsctx->validregs |= SCE_HIGS_VALID_FRAME | SCE_HIGS_VALID_ZBUF
                  | SCE_HIGS_VALID_TEST;
```

When isZbuf==0

```
gsctx->isZbuf=0;
gsctx->value.zbuf.PSM=0;
gsctx->value.zbuf.ZMSK=1;
gsctx->value.test.ZTE=1;
gsctx->value.test.ZTST=SCE_GS_ZALWAYS;
```

When isZbuf==1

```
gsctx->value.zbuf.PSM=zpsm&0xff;
```

```
gsctx->value.zbuf.ZMSK=0;
```

```
gsctx->value.test.ZTE=1;
```

```
gsctx->value.test.ZTST=SCE_GS_ZGREATER;
```

Values that are not set here within frame, zbuf, and test registers

frame: FBP, FBW, FBMSK (the initial value is also set for FBMSK when FBMSK is created)

zbuf: ZBP

test: ATE, ATST, AREF, AFAIL, DATE, DATM (initial values are also set for everything other than ATE when they are created)

Return value

Table 5-31

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Argument is invalid

sceHiGsCtxSetEachBuffer

Allocate GS memory

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSetEachBuffer(

sceHiGsCtx *gsctx,	Pointer to management structure of port for which buffers are to be set
u_int fbp0,	GS memory address to be set for fbp[0] (units: words/2048)
u_int fbp1,	GS memory address to be set for fbp[1] (units: words/2048)
u_int zbp)	GS memory address to be set for value.zbuf.ZBP (units: word/2048)

Calling conditions

None

Description

This function allocates the given GS memory address to each buffer pointer.

Notes

The setting contents are as follows. Since transfers of the frame register and zbuf register are enabled, confirm that values other than the buffer pointers are set correctly in these registers.

```
sctx->fbp[0]=fbp0;
sctx->fbp[1]=fbp1;
sctx->value.zbuf.ZBP=zbp;
```

```
sctx->validregsl=SCE_HIGS_VALID_FRAMEISCE_HIGS_VALID_ZBUF;
```

Return value

Table 5-32

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetFrame

Set FRAME register and frame buffer pointers

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetFrame(**

sceHiGsCtx <i>*gsctx</i> ,	Pointer to management structure of port for which values are to be set
int <i>fbp0</i> ,	Buffer pointer to be set for management structure fbp[0]
int <i>fbp1</i> ,	Buffer pointer to be set for management structure fbp[1]
int <i>fbw</i> ,	FRAME register fbw
int <i>psm</i> ,	FRAME register psm
int <i>fbmsk</i>)	FRAME register fbmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets values in the port's register setting area and enables the Frame register's transfer setting. When the port has a single buffer setting, specify a dummy value for *fbp1*.

Return value

Table 5-33

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetHalfOffset

Set drawing offset data for interlace

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4.3	March 29, 2002

Syntax

```
sceHiErr sceHiGsCtxSetHalfOffset(
    sceHiGsCtx *gsctx,           Pointer to management structure of port for which
                                value is to be set
    int field)                   Offset increment control
                                (0: Do not increment; 1: Increment)
```

Calling conditions

None

Description

When *field* is 1, this function adds 8 to the Y-direction drawing offset of XYOFFSET for the setting area and sets it in the transfer area. When *field* is 0, this function sets the transfer area without incrementing the offset.

At the same time, the value of the assigned field is set in the *field* element of the port's management structure.

This data is set regardless of the value of the isInterlace flag in the management structure.

This function should be called after calling sceHiGsCtxUpdate().

Since this function changes the transfer area directly, it is not necessary to call sceHiGsCtxUpdate() after calling this function.

Return value

Table 5-34

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetLumpBuffer

Allocate GS memory

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSetLumpBuffer(
 sceHiGsCtx *gsctx,

 u_int bp)

Pointer to management structure of port for which
value is to be set

Address of GS memory to be allocated (units:
words/2048)

Calling conditions

None

Description

This function allocates memory to each buffer from the given GS memory in the order fbp[1], fbp[0], and value.zbuf.ZBP when a double buffer is used and in the order fbp[0] and value.zbuf.ZBP when a single buffer is used.

Notes

This function enables frame register and zbuf register transfers. As a result, confirm that values other than buffer pointers are set correctly within these registers.

Return value

Table 5-35

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRect

Set screen size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetRect(****sceHiGsCtx** *gsctx,

Pointer to management structure of port for which value is to be set

int x,

Offset from left side of screen (units: pixels)

int y,

Offset from top of screen (units: pixels)

int w,

Screen width (units: pixels)

int h,

Screen height (units: pixels)

int fbw)

Frame buffer width

SCE_HIGS_FBW_DEFAULT Calculate minimum buffer width from x and w

SCE_HIGS_FBW_KEEP Do not change current value.

Other digits Sets specified value (units: pixels: 64)

Calling conditions

None

Description

This function sets the screen size of the port with a rectangle. The registers that are set are the FBW bit fields of the xyoffset, scissor, and frame registers. This function also sets the sprite vertex of the clear transfer area.

Notes

The setting contents are as follows.

```
gsctx->validregs |= SCE_HIGS_VALID_FRAME | SCE_HIGS_VALID_XYOFFSET
| SCE_HIGS_VALID_SCISSOR;
```

```
gsctx->value.xyoffset.OFX = (2048-(w/2)-x)*16;
```

```
gsctx->value.xyoffset.OFY = (2048-(h/2)-y)*16;
```

```
gsctx->value.scissor.SCAX0 = x;
```

```
gsctx->value.scissor.SCAX1 = x+w-1;
```

```
gsctx->value.scissor.SCAY0 = y;
```

```
gsctx->value.scissor.SCAY1 = y+h-1;
```

```

if (fbw==SCE_HIGS_FBW_DEFAULT) gsctx->value.frame.FBW = (x+w+63)/
64;
else if (fbw==SCE_HIGS_FBW_KEEP) gsctx->value.frame.FBW = gsctx->
value.
frame.FBW;

gsctx->clearp.clear.xyz2a.X = gsctx->value.xyoffset.OFX+(x<<4);
gsctx->clearp.clear.xyz2a.Y = gsctx->value.xyoffset.OFY+(y<<4);

gsctx->clearp.clear.xyz2b.X = gsctx->value.xyoffset.OFX + ((x+w)<
<4);
gsctx->clearp.clear.xyz2b.Y = gsctx->value.xyoffset.OFY + ((y+h)<
<4);

```

Return value

Table 5-37

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegAlpha

Set ALPHA register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr** sceHiGsCtxSetRegAlpha(**sceHiGsCtx** *gsctx,,

Pointer to management structure of port for which value is to be set

int a,

ALPHA register a

int b,

ALPHA register b

int c,

ALPHA register c

int d,

ALPHA register d

int fix)

ALPHA register fix

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area, and enables the Alpha register's transfer setting.

Return value

Table 5-38

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegClamp

Set CLAMP register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiErr sceHiGsCtxSetRegClamp(
    sceHiGsCtx *gsctx,,           Pointer to management structure of port for which
                                   value is to be set
    int wms,                      CLAMP register wms
    int wmt,                      CLAMP register wmt
    int minu,                    CLAMP register minu
    int maxu,                    CLAMP register maxu
    int minv,                    CLAMP register minv
    int maxv)                    CLAMP register maxv
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Clamp register's transfer setting.

Return value

Table 5-39

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegFba

Set FBA register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetRegFba(****sceHiGsCtx** *gsctx,,Pointer to management structure of port for which
value is to be set**int** fba)

FBA register fba

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Fba register's transfer setting.

Return value

Table 5-40

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegMiptbp1

Set MIPTBP1 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetRegMiptbp1(**

sceHiGsCtx *gsctx,,	Pointer to management structure of port for which value is to be set
int tbp1,	MIPTBP1 register tbp1
int tbw1,	MIPTBP1 register tbw1
int tbp2,	MIPTBP1 register tbp2
int tbw2,	MIPTBP1 register tbw2
int tbp3,	MIPTBP1 register tbp3
int tbw3)	MIPTBP1 register tbw3

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Miptbp1 register's transfer setting.

Return value

Table 5-41

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegMiptbp2

Set MIPTBP2 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr** sceHiGsCtxSetRegMiptbp2(

sceHiGsCtx *gsctx,,	Pointer to management structure of port for which value is to be set
int tbp4,	MIPTBP2 register tbp4
int tbw4,	MIPTBP2 register tbw4
int tbp5,	MIPTBP2 register tbp5
int tbw5,	MIPTBP2 register tbw5
int tbp6,	MIPTBP2 register tbp6
int tbw6)	MIPTBP2 register tbw6

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Miptbp2 register's transfer setting.

Return value

Table 5-42

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegTest

Set TEST register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsTestRegs(

sceHiGsCtx *gsctx,,	Pointer to management structure of port for which value is to be set
int ate,	TEST register ate
int atst,	TEST register atst
int aref,	TEST register aref
int afail,	TEST register afail
int date,	TEST register date
int datm,	TEST register datm
int zte,	TEST register zte
int ztst)	TEST register ztst

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Test register's transfer setting.

Return value

Table 5-43

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegTex0

Set TEX0 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetRegTex0(**

sceHiGsCtx <i>*gsctx</i> ,	Pointer to management structure of port for which value is to be set
int <i>tbp0</i> ,	TEX0 register tbp0
int <i>tbw</i> ,	TEX0 register tbw
int <i>psm</i> ,	TEX0 register psm
int <i>tw</i> ,	TEX0 register tw
int <i>th</i> ,	TEX0 register th
int <i>tcc</i> ,	TEX0 register tcc
int <i>tfx</i> ,	TEX0 register tfx
int <i>cbp</i> ,	TEX0 register cbp
int <i>cpsm</i> ,	TEX0 register cpsm
int <i>csm</i> ,	TEX0 register csm
int <i>csa</i> ,	TEX0 register csa
int <i>cl</i>)	TEX0 register cl

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Tex0 register's transfer setting.

Return value

Table 5-44

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegTex1

Set TEX1 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetRegTex1(**

sceHiGsCtx *gsctx,,	Pointer to management structure of port for which value is to be set
int lcm,	TEX1 register lcm
int mxl,	TEX1 register mxl
int mmag,	TEX1 register mmag
int mmin,	TEX1 register mmin
int mtba,	TEX1 register mtba
int l,	TEX1 register l
int k)	TEX1 register k

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Tex1 register's transfer setting.

Return value

Table 5-45

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegXyoffset

Set XYOFFSET register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr** sceHiGsCtxSetRegXyoffset(**sceHiGsCtx** *gsctx,,Pointer to management structure of port for which
value is to be set**int** ofx,

XYOFFSET register ofx

int ofy)

XYOFFSET register ofy

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Xyoffset register's transfer setting.

Return value

Table 5-46

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegZbuf

Set ZBUF register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetRegZbuf(**

sceHiGsCtx <i>*gsctx</i> ,	Pointer to management structure of port for which value is to be set
int <i>zbp</i> ,	ZBUF register fbp
int <i>psm</i> ,	ZBUF register psm
int <i>zmsk</i>)	ZBUF register zmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area and enables the Zbuf register's transfer setting.

Return value

Table 5-47

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetZbufDepth

Set Z-buffer format

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxSetZbufDepth(****sceHiGsCtx** *gsctx,

Pointer to management structure of port for which value is to be set

int zpsm,

Z-buffer format

int isZbuf)

1: Use Z-buffer

0: Do not use Z-buffer

Calling conditions

None

Description

This function sets the format of the Z-buffer.

If the Z-buffer will not be used, zpsm should be set to a dummy value.

Example 1: When 24-bit Z-buffer is to be used

```
sceHiGsCtxSetZbufDepth(gsctx, SCE_GS_PSMZ24, 1);
```

Example 2: When a Z-buffer will not be used

```
sceHiGsCtxSetZbufDepth(gsctx, 0, 0);
```

Notes

The setting contents are as follows. Since transfers are registered for the zbuf and test registers, either set missing settings separately within these registers or cancel the transfer registration.

```
gsctx->isZbuf=isZbuf
```

```
gsctx->validregsl=SCE_HIGS_VALID_ZBUFISCE_HIGS_VALID_TEST;
```

When isZbuf==0

```
gsctx->value.zbuf.PSM=0;
```

```
gsctx->value.zbuf.ZMSK=1;
```

```
gsctx->value.test.ZTE=1;
```

```
gsctx->value.test.ZTST=SCE_GS_ZALWAYS;
```

When isZbuf==1

```
gsctx->value.zbuf.PSM=zpsm&0xff;
```

```
gsctx->value.zbuf.ZMSK=0;
```

```
gsctx->value.test.ZTE=1;
```



```
gsctx->value.test.ZTST=SCE_GS_ZGREATER;
```

Values that are not set here within zbuf and test registers

zbuf: ZBP

test: ATE, ATST, AREF, AFAIL, DATE, DATM (initial values are also set for everything other than ATE when they are created)

Return value

Table 5-48

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Argument is invalid

sceHiGsCtxShiftBuffers

Shift GS memory assignment

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsCtxShiftBuffers(****sceHiGsCtx** *gsctx,Pointer to management structure of port for which
value is to be set**int** fbpoffset)

Shift amount (units: words/2048)

Calling conditions

None

Description

This function shifts the buffer pointers that were set for the specified port by the amount fbpoffset.

Notes

The setting contents are as follows. Since transfers of the frame register and zbuf register are enabled, confirm that values other than the buffer pointers are also set correctly within these registers.

```
gsctx->validregs|=SCE_HIGS_VALID_FRAME|SCE_HIGS_VALID_ZBUF;
```

```
gsctx->fbp[0]+=fbpoffset;
```

```
gsctx->fbp[1]+=fbpoffset;
```

```
gsctx->value.zbuf.ZBP+=fbpoffset;
```

Return value

Table 5-49

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSwap

Swap double buffer and set interlace

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSwap(

sceHiGsCtx *gsctx,

Pointer to management structure of port for which value is to be set

int swap,

Buffer ID for double buffer

int field)

Field ID for interlace (top field: 0, bottom field: 1)

Calling conditions

None

Description

This function swaps the double buffer and sets the value of the interlace offset.

It sets the value of *field* in the *field* member of the management structure.

When the relevant port has a double buffer setting, the value of (swap&1) is set in the *swap* member.

When a double buffer is specified and frame register transfer is enabled, the value of the fbp[swap&1] member is set as the value of the frame buffer pointer value in the transfer area.

This function differs from other setting functions in that the sceHiGsCtxUpdate() function need not be called after this function is called because this function updates the transfer area directly.

Return value

Table 5-50

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSwapAll

Swap double buffer and set interlace (for all synchronous ports of the context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxSwapAll(

int <i>swap</i> ,	Buffer ID for double buffer
int <i>field</i>)	Field ID for interlace (top field: 0, bottom field: 1)

Calling conditions

None

Description

This function calls the `sceHiGsCtxSwap()` function to swap the double buffer and set the value of the interlace offset for those ports for which the synchronous flag (`isSync`) is 1 among all of the ports within the context register group.

This function differs from other setting functions in that the `sceHiGsCtxUpdate()` function need not be called after this function is called because this function updates the transfer area directly.

Return value

Table 5-51

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxUpdate

Update transfer area (context register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsCtxUpdate(

sceHiGsCtx *gsctx)

Pointer to management structure of port for which value is to be updated

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function reflects the value that was set for the port's setting area in the transfer area.

In addition to changing the port's setting, this function must be called before the area is transferred (excluding (sceHiGsCtxSwap())).

Return value

Table 5-52

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvCopy

Copy port (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvCopy(****sceHiGsEnv** *dst, Pointer to management structure of destination port**sceHiGsEnv** *src) Pointer to management structure of source port**Calling conditions**

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function makes a copy of the contents of the port. Copying can be performed only between ports having the same transfer register settings.

If an attempt is made to copy a port having a different transfer register setting, it will not be copied and an error will be returned.

Since the transfer area is not copied, after this function is called, the transfer area must be updated by calling the sceHiGsEnvUpdate() function before the area is transferred.

Return value

Table 5-53

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to copy a port having a different transfer register setting

sceHiGsEnvCreate

Create port (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

```
sceHiGsEnv *sceHiGsEnvCreate(
    u_int validregs)           Register list to be transferred by this port
```

Calling conditions

None

Description

This function acquires and initializes one environment register group management structure and returns a pointer to it.

The register list is specified according to the logical OR of the following declaration bits.

Register list specification:

colclamp register	SCE_HIGS_VALID_COLCLAMP
dimx register	SCE_HIGS_VALID_DIMX
dthe register	SCE_HIGS_VALID_DTHE
fog register	SCE_HIGS_VALID_FOG
fogcol register	SCE_HIGS_VALID_FOGCOL
pabe register	SCE_HIGS_VALID_PABE
texa register	SCE_HIGS_VALID_TEXA
prmode register	SCE_HIGS_VALID_PRMODE
prmodecont register	SCE_HIGS_VALID_PRMODECONT
texclut register (not implemented)	SCE_HIGS_VALID_TEXCLUT
scanmsk register (not implemented)	SCE_HIGS_VALID_SCANMSK
texflush register	SCE_HIGS_VALID_TEXFLUSH
bitblt register (not implemented)	SCE_HIGS_VALID_BITBLT
trxpos register (not implemented)	SCE_HIGS_VALID_TRXPOS
trxreg register (not implemented)	SCE_HIGS_VALID_TRXREG
trxdir register (not implemented)	SCE_HIGS_VALID_TRXDIR
signal register (not implemented)	SCE_HIGS_VALID_SIGNAL
finish register	SCE_HIGS_VALID_FINISH
label register (not implemented)	SCE_HIGS_VALID_LABEL

Example: When specifying the texa and pabe registers

```
sceHiGsEnv *port;
port=sceHiGsEnvCreate(SCE_HIGS_VALID_PABE|SCE_HIGS_VALID_TEXA);
if (port==NULL) (error processing)
```

Notes

The initial values are set as follows. (Only registers that were specified by arguments are valid.)

COLCLAMP: CLAMP = 1

PRMODECONT: AC = 1

TEXA: TA0=127

AEM=1

TA1=128

TRXDIR: XDR=3

DTHE, PRMODE, DIMX, FOG, FOGCOL, PABE, TEXCLUT, SCANMSK, TEXFLUSH,

BITBLT, TRXPOS, TRXREG, SIGNAL, FINISH, LABEL: All 0

For unimplemented registers (TEXCLUT, SCANMSK, BITBLT, TRXPOS, TRXREG, TRXDIR, SIGNAL, and LABEL), the values cannot be changed from the initial values (because the setting functions are not implemented). These can be used as long as only the initial value is used.

Return value

When processing is successful, a pointer to the management structure of the created port is returned.

When there is an error, the following is returned:

Table 5-54

Value	Result
NULL	Internal memory area allocation failed

sceHiGsEnvDelete

Free port (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsEnvDelete(

sceHiGsEnv *gsenv) Pointer to management structure of port to be released

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function frees the memory used by the port.

Return value

Table 5-55

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvFcache

Flush transfer area cache (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvFcache(****sceHiGsEnv** *gsenv)Pointer to management structure of port for which
cache is to be flushed**Calling conditions**

Cannot be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function flushes the transfer area cache of the specified port.

This function is called when sceHiGsEnvSend() is used for a DMA transfer immediately after the transfer area was updated by sceHiGsEnvUpdate().

Notes

sceHiGsEnvFcache() should be used within the interrupt handler.

Return value

Table 5-56

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvFcacheI

Flush transfer area cache (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsEnvFcacheI(

sceHiGsEnv *gsenv)

Pointer to management structure of port for which
cache is to be flushed

Calling conditions

Can be called from interrupt handler

Cannot be called from a thread

Description

This function flushes the transfer area cache of the specified port. This function is called when sceHiGsEnvSend() is used for a DMA transfer immediately after the transfer area was updated by sceHiGsEnvUpdate().

Notes

This function is for use within the sceHiGsEnvFcache() interrupt handler.

Return value

Table 5-57

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvGetDefault

Get default port (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiGsEnv *sceHiGsEnvGetDefault(void)

Calling conditions

None

Description

This function gets the default port that was set with the sceHiGsEnvSetDefault() function.

Return value

Returns a pointer to the management structure of the environment register group's default port.

sceHiGsEnvRegist

Register transfer (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsEnvRegist(

sceHiGsEnv *gsenv)

Pointer to management structure of port for which transfer is to be registered

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function uses the transfer registration function to register a transfer of the transfer area for the specified port.

The transfer registration function is set with `sceHiGsServiceSetRegistFunc()`.

By default, since the transfer is registered as an HiG DMA library dynamic chain, the actual transfer will occur when `sceHiDMASend()` is called later.

Notes

Whether or not multithreading can be used depends on the transfer registration function. Since in the initial state the registered function is not multithread safe, this function is also not multithread safe. However, if the registered function was multithread safe, this function will be multithread safe.

Return value

Table 5-58

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	Transfer registration function returned an error

sceHiGsEnvSend

Transfer immediately (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsEnvSend(

sceHiGsEnv *gsenv)

Pointer to management structure of port for which transfer area is to be transferred

Calling conditions

None

Description

This function uses the DMA ch.2 normal mode for GIF transfers to immediately transfer the transfer area of the specified port. Before calling this function, you must confirm that the transfer area cache is flushed and that DMA ch.2 is available. This function completes without confirming the end of the last DMA transfer.

Return value

Table 5-59

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvSetDefault

Set default port (environment register group)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsEnvSetDefault(

sceHiGsEnv *gsenv)

Pointer to management structure to be specified for default port

Calling conditions

None

Description

This function sets *gsenv* in the environment register group's default port.

Return value

Table 5-60

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvSetRegBitblt

Set Bitblt register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegBitblt(**

sceHiGsEnv *gsenv, ,	Pointer to management structure of port for which values are to be set
int sbp,	BITBLT register sbp
int sbw,	BITBLT register sbw
int spsm,	BITBLT register spsm
int dbp,	BITBLT register dbp
int dbw,	BITBLT register dbw
int dpsm)	BITBLT register dpsm

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets values in the port's register setting area. If the BITBLT register's transfer setting is invalid, this function returns an error.

Return value

Table 5-61

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvSetRegColclamp

Set COLCLAMP register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegColclamp(****sceHiGsEnv** *gsenv, ,Pointer to management structure of port for which
value is to be set**int** clamp)

COLCLAMP register clamp

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the Colclamp register's transfer setting is disabled, an error is returned.

Return value

Table 5-62

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegDimx

Set DIMX register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegDimx(****sceHiGsEnv** *gsenv, ,Pointer to management structure of port for which
value is to be set**int** dm[16])

DIMX register dm

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the Dimx register's transfer setting is disabled, an error is returned.

Return value

Table 5-63

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegDthe

Set DTHE register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr** sceHiGsEnvSetRegDthe(**sceHiGsEnv** *gsenv,Pointer to management structure of port for which
value is to be set**int** dthe)

DTHE register dthe

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the *dthe* register's transfer setting is disabled, an error is returned.

Return value

Table 5-64

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegFog

Set FOG register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegFog(****sceHiGsEnv** *gsenv, ,Pointer to management structure of port for which
value is to be set**int** f)

FOG register f

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the Fog register's transfer setting is disabled, an error is returned.

Return value

Table 5-65

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegFogcol

Set FOGCOL register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsEnvSetRegFogcol(

sceHiGsEnv *gsenv, , Pointer to management structure of port for which value is to be set

int fcr, FOGCOL register fcr

int fcg, FOGCOL register fcg

int fcb) FOGCOL register fcb

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the Fogcol register's transfer setting is disabled, an error is returned.

Return value

Table 5-66

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegLabel

Set Label register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegLabel(**

sceHiGsEnv *gsenv, Pointer to management structure of port for which values are to be set

u_int id, LABEL register id

u_int idmsk) LABEL register idmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets values in the port's register setting area. If the LABEL register's transfer setting is invalid, this function returns an error.

Return value

Table 5-67

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegPabe

Set PABE register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr** sceHiGsEnvSetRegPabe(**sceHiGsEnv** *gsenv, ,Pointer to management structure of port for which
value is to be set**int** pabe)

PABE register pabe

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the *pabe* register's transfer setting is disabled, an error is returned.

Return value

Table 5-68

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegPrmode

Set PRMODE register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegPrmode(**

sceHiGsEnv *gsenv, ,	Pointer to management structure of port for which value is to be set
int iip,	PRMODE register iip
int tme,	PRMODE register tme
int fge,	PRMODE register fge
int abe,	PRMODE register abe
int aa1,	PRMODE register aa1
int fst,	PRMODE register fst
int ctxt,	PRMODE register ctxt
int fix)	PRMODE register fix

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the Prmode register's transfer setting is disabled, an error is returned.

Return value

Table 5-69

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegPrmodecont

Set PRMODECONT register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegPrmodecont(****sceHiGsEnv** *gsenv, ,Pointer to management structure of port for which
value is to be set**int** ac)

PRMODECONT register ac

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the Prmodecont register's transfer setting is disabled, an error is returned.

Return value

Table 5-70

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegScanmsk

Set Scanmsk register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegScanmsk(****sceHiGsEnv *gsenv, ,**Pointer to management structure of port for which
value is to be set**int msk)**

SCANMSK register msk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. If the SCANMSK register's transfer setting is invalid, this function returns an error.

Return value

Table 5-71

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegSignal

Set Signal register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax

```
sceHiErr sceHiGsEnvSetRegSignal(
    sceHiGsEnv *gsenv, ,           Pointer to management structure of port for which
                                   values are to be set
    u_int id,                       SIGNAL register id
    u_int idmsk)                    SIGNAL register idmsk
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets values in the port's register setting area. If the SIGNAL register's transfer setting is invalid, this function returns an error.

Return value

Table 5-72

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegTexa

Set TEXA register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegTexa(**

sceHiGsEnv *gsenv, , Pointer to management structure of port for which value is to be set

int ta0, TEXA register ta0

int aem, TEXA register aem

int ta1) TEXA register ta1

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. When the Texa register's transfer setting is disabled, an error is returned.

Return value**Table 5-73**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegTexclut

Set Texclut register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax

sceHiErr sceHiGsEnvSetRegTexclut(

sceHiGsEnv *gsenv, , Pointer to management structure of port for which values are to be set

int cbw, TEXCLUT register cbw

int cou, TEXCLUT register cou

int cov) TEXCLUT register cov

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets values in the port's register setting area. If the TEXCLUT register's transfer setting is invalid, this function returns an error.

Return value

Table 5-74

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegTrxdir

Set Trxdir register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax**sceHiErr** sceHiGsEnvSetRegTrxdir(**sceHiGsEnv** *gsenv, ,Pointer to management structure of port for which
value is to be set**int** xdr)

TRXDIR register xdr

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the port's register setting area. If the TRXDIR register's transfer setting is invalid, this function returns an error.

Return value

Table 5-75

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegTrxpos

Set Trxpos register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax**sceHiErr sceHiGsEnvSetRegTrxpos(**

sceHiGsEnv *gsenv,	Pointer to management structure of port for which values are to be set
int ssax,	TRXPOS register ssax
int ssay,	TRXPOS register ssay
int dsax,	TRXPOS register dsax
int dsay,	TRXPOS register dsay
int dir)	TRXPOS register dir

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets values in the port's register setting area. If the TRXPOS register's transfer setting is invalid, this function returns an error.

Return value

Table 5-76

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvSetRegTrxreg

Set Trxreg register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.4	March 29, 2002

Syntax**sceHiErr** sceHiGsEnvSetRegTrxreg(**sceHiGsEnv** *gsenv, ,

Pointer to management structure of port for which values are to be set

int rrw,

TRXREG register rrw

int rrh)

TRXREG register rrh

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets values in the port's register setting area. If the TRXREG register's transfer setting is invalid, this function returns an error.

Return value

Table 5-77

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
4	SCE_HIG_FAILURE	An attempt was made to set the register with disabled transfer setting

sceHiGsEnvUpdate

Update transfer area of environment register group port

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsEnvUpdate(

sceHiGsEnv *gsenv)

Pointer to management structure of port for which value is to be updated

Calling conditions

None

Description

This function reflects the value that was set for the port's setting area in the transfer area.

In addition to changing the port's setting, this function must be called before the area is transferred.

Return value

Table 5-78

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Number of transfer registers has been incorrectly changed

sceHiGsServiceExit

Exit GS register management service

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsServiceExit(void)****Calling conditions**

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function performs termination processing for the entire GS register management service. It also frees the standard port of each register group.

Normally, this function is called within the sceHiGsDisplayEnd() function. However, when that library function is not used, this function should be called separately to exit the GS register management service.

Return value

Table 5-79

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsServiceInit

Initialize GS register management service

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax

sceHiErr sceHiGsServiceInit(void)

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function initializes the entire GS register management service.

It also acquires and initializes each register group's standard port and sets it as the register group's default port.

Normally, this function is called within the sceHiGsDisplaySize() function. However, when that library function is not used, this function should be called before using the GS register management service. In this case, since the context register group's standard port sceHiGsStdCtx setting is not suitable, you should set a suitable correct value.

This function cannot be called again without calling sceHiGsServiceExit().

Return value

Table 5-80

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
4	SCE_HIG_FAILURE	sceHiGsServiceInit() has already been called

sceHiGsServiceSetRegistFunc

Set function to be used for transfer registration

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.3	March 29, 2002

Syntax**sceHiErr sceHiGsServiceSetRegistFunc(****int (*func)(void *, int)**

Function to be used for transfer registration

Calling conditions

None

Description

The function to be called when registering a transfer within the sceHiGsCtxRegist() or sceHiGsEnvRegist() function can be set here. The function that was set is called with the following arguments.

func(addr, size);

void *addr

int size

addr Starting address of transfer area (giftag with eop=1 is appended at the beginning)

size Size to be transferred (units: Qwords)

When the transfer registration succeeds, the setting function must return zero and when it fails, the setting function must return a non-zero value.

Note that when the sceHiGsCtxRegist() function is used to register a transfer with a clear, the transfer registration is performed twice.

When NULL is assigned to *func*, the default transfer registration function (which uses a HiG DMA packet management service dynamic chain) will be set.

Return value

Table 5-81

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

Old GS Register Setting Functions

sceHiGsAlphaRegs

Set ALPHA register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsAlphaRegs(**

int *a*,

int *b*,

int *c*,

int *d*,

int *fix*)

ALPHA register a

ALPHA register b

ALPHA register c

ALPHA register d

ALPHA register fix

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

sceHiGsCtxSetRegAlpha(sceHiGsStdCtx, a, b, c, d, fix);

Return value

Table 5-82

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsClampRegs

Set CLAMP register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsClampRegs(**

int <i>wms</i> ,	CLAMP register wms
int <i>wmt</i> ,	CLAMP register wmt
int <i>minu</i> ,	CLAMP register minu
int <i>maxu</i> ,	CLAMP register maxu
int <i>minv</i> ,	CLAMP register minv
int <i>maxv</i>)	CLAMP register maxv

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegClamp (sceHiGsStdCtx, wms, wmt, minu, maxu, minv,maxv);
```

Return value

Table 5-83

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsClear

Set clear

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```
sceHiErr sceHiGsClear(
    u_int mode)                Mode specification when clearing
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the mode to be used when clearing.

mode is set to a value of the enumerated type sceHiGsClear_t.

If the value is not SCE_HIGS_CLEAR_ALL, this function performs the same processing as AFail for the TEST register.

This function sets the clear mode of the context group's standard port.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetClearMode(sceHiGsStdCtx, mode);
```

Return value

Table 5-84

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsClearColor

Set clear color

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsClearColor(

u_char <i>red</i> ,	R value
u_char <i>green</i> ,	G value
u_char <i>blue</i> ,	B value
u_char <i>alpha</i>)	A value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the clear color.

It sets a value in the register setting area of the context group's standard port.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetClearColor(sceHiGsStdCtx, red, green, blue, alpha);
```

Return value

Table 5-85

Value	Macro	Result
00	SCE_HIG_NO_ERR	Processing was successful

sceHiGsClearDepth

Set clear depth

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsClearDepth(
 u_int z, **Z-value**

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the clear depth.

It sets the clear Z-value of the context group's standard port.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

 sceHiGsCtxSetClearZ(sceHiGsStdCtx, z);

Return value

Table 5-86

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsColclampRegs

Set COLCLAMP register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```
sceHiErr sceHiGsColclampRegs(
    int clamp)                COLCLAMP register clamp
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegColclamp (sceHiGsStdCtx, clamp);
```

Return value

Table 5-87

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsContextID

Switch context

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsContextID(
int id) Context ID

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function switches the current context.
When id=0, context 1 is used. When id=1, context 2 is used.
If an id other than 0 or 1 is specified, an error is returned.
This function is retained for compatibility with previous libraries.
It has been replaced by the following function:
sceHiGsCtxSetContext(sceHiGsStdCtx, id);

Return value

Table 5-88

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	Argument is invalid

sceHiGsContextStatus

Get current context

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiGsContext *sceHiGsContextStatus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns a pointer to the register setting area of the context group's standard port.

This function is retained for compatibility with previous libraries.

Return value

Returns an sceHiGsContext pointer.

sceHiGsDtheRegs

Set DTHE register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsDtheRegs(

int <i>dthe</i>)	DTHE register dthe
--------------------------	--------------------

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegDthe (sceHiGsStdCtx, dthe);
```

Return value

Table 5-90

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsFbaRegs

Set FBA register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```
sceHiErr sceHiGsFbaRegs(
    int fba)                FBA register fba
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegFba (sceHiGsStdCtx, fba);
```

Return value

Table 5-91

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsFogcolRegs

Set FOGCOL register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsFogcolRegs(

int <i>fcn</i> ,	FOGCOL register fcn
int <i>fcg</i> ,	FOGCOL register fcg
int <i>fcbl</i>)	FOGCOL register fcbl

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegFogcol (sceHiGsStdCtx, fcr, fcg, fcb);
```

Return value

Table 5-92

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsFogRegs

Set FOG register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```
sceHiErr sceHiGsFogRegs(
    int f)                                FOG register f
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegFog (sceHiGsStdCtx, f);
```

Return value

Table 5-93

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsFrameRegs

Set FRAME register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsFrameRegs(**

int <i>fbp</i> ,	FRAME register fbp
int <i>fbw</i> ,	FRAME register fbw
int <i>psm</i> ,	FRAME register psm
int <i>fbmsk</i>)	FRAME register fbmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetFrame (sceHiGsStdCtx, fbp0, fbp1, fbw, psm, fbmsk);
```

For the settings of fbp0 and fbp1, see the reference for the sceHiGsSetFrame() function.

Return value**Table 5-94**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsGeneralStatus

Get general-purpose register contents

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiGsGeneral *sceHiGsGeneralStatus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns a pointer to the structure (register setting area of environment group's standard port) where the contents of internally managed general-purpose registers are kept.

This function is retained for compatibility with previous libraries.

Return value

Returns an sceHiGsGeneral pointer.

sceHiGsMiptbp1Regs

Set MIPTBP1 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsMiptbp1Regs(**

int <i>tbp1</i> ,	MIPTBP1 register tbp1
int <i>tbw1</i> ,	MIPTBP1 register tbw1
int <i>tbp2</i> ,	MIPTBP1 register tbp2
int <i>tbw2</i> ,	MIPTBP1 register tbw2
int <i>tbp3</i> ,	MIPTBP1 register tbp3
int <i>tbw3</i>)	MIPTBP1 register tbw3

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegMiptbp1 (sceHiGsStdCtx, tbp4, tbw4, tbp5, tbw5, tbp6, tbw6);
```

Return value

Table 5-95

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsMiptbp2Regs

Set MIPTBP2 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsMiptbp2Regs(

int <i>tbp4</i> ,	MIPTBP2 register tbp4
int <i>tbw4</i> ,	MIPTBP2 register tbw4
int <i>tbp5</i> ,	MIPTBP2 register tbp5
int <i>tbw5</i> ,	MIPTBP2 register tbw5
int <i>tbp6</i> ,	MIPTBP2 register tbp6
int <i>tbw6</i>)	MIPTBP2 register tbw6

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegMiptbp2 (sceHiGsStdCtx, pabe);
```

Return value

Table 5-96

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsPabeRegs

Set PABE register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```
sceHiErr sceHiGsPabeRegs(
    int pabe)                PABE register pabe
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegPabe (sceHiGsStdCtx, sceHiGsPacked *p);
```

Return value

Table 5-97

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsPackedDelete

Delete PACKED data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsPackedDelete(
sceHiGsPacked *p) Pointer to PACKED data to be deleted

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

You should use the GS register management service function instead.

Return value

Table 5-99

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsPackedUpdate

Update PACKED data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsPackedUpdate(
sceHiGsPacked *p) Pointer to PACKED data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function updates the current contents of the library's internally managed GS general-purpose register variables (including context) in the data of p.

A suitable register address must be entered in advance for the addr of p.

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

You should use the GS register management service function instead.

Return value

Table 5-100

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	Argument is invalid

sceHiGsPrmodecontRegs

Set PRMODECONT register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

```
sceHiErr sceHiGsPrmodecontRegs(
    int ac)                PRMODECONT register ac
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value for the library's internally managed GS general-purpose register variables.

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegPrmodecont (sceHiGsStdCtx, ac);
```

Return value**Table 5-101**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsPrmodeRegs

Set PRMODE register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsPrmodeRegs(

int <i>iip</i> ,	PRMODE register iip
int <i>tme</i> ,	PRMODE register tme
int <i>fge</i> ,	PRMODE register fge
int <i>abe</i> ,	PRMODE register abe
int <i>aa1</i> ,	PRMODE register aa1
int <i>fst</i> ,	PRMODE register fst
int <i>ctxt</i> ,	PRMODE register ctxt
int <i>fix</i>)	PRMODE register fix

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegPrmode (sceHiGsStdCtx, iip, tme, fge, abe, aa1, fst, ctxt, fix);
```

Return value

Table 5-102

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsTestRegs

Set TEST register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsTestRegs(**

int <i>ate</i> ,	TEST register ate
int <i>atst</i> ,	TEST register atst
int <i>aref</i> ,	TEST register aref
int <i>afail</i> ,	TEST register afail
int <i>date</i> ,	TEST register date
int <i>datm</i> ,	TEST register datm
int <i>zte</i> ,	TEST register zte
int <i>ztst</i>)	TEST register ztst

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetRegTest (sceHiGsStdCtx, ate, atst, aref, afail, date, datm, zte, ztst);
```

Return value

Table 5-103

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsTex0Regs

Set TEX0 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsTex0Regs(

int <i>tbp0</i> ,	TEX0 register tbp0
int <i>tbw</i> ,	TEX0 register tbw
int <i>psm</i> ,	TEX0 register psm
int <i>tw</i> ,	TEX0 register tw
int <i>th</i> ,	TEX0 register th
int <i>tcc</i> ,	TEX0 register tcc
int <i>tfx</i> ,	TEX0 register tfx
int <i>cbp</i> ,	TEX0 register cbp
int <i>cpsm</i> ,	TEX0 register cpsm
int <i>csm</i> ,	TEX0 register csm
int <i>csa</i> ,	TEX0 register csa
int <i>cld</i>)	TEX0 register cld

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

sceHiGsCtxSetReg Tex0 (sceHiGsStdCtx, tbp0, tbw, psm, tw, th, tcc, tfx, cbp, cpsm, csm, csa, cld);

Return value

Table 5-104

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsTex1Regs

Set TEX1 register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsTex1Regs(**

int <i>lcm</i> ,	TEX1 register lcm
int <i>mxl</i> ,	TEX1 register mxl
int <i>mmag</i> ,	TEX1 register mmag
int <i>mmin</i> ,	TEX1 register mmin
int <i>mtba</i> ,	TEX1 register mtba
int <i>l</i> ,	TEX1 register l
int <i>k</i>)	TEX1 register k

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

```
sceHiGsCtxSetReg Tex1 (sceHiGsStdCtx, lcm, mxl, mmag, mmin, mtba, l, k);
```

Return value

Table 5-105

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsTexaRegs

Set TEXA register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsTexaRegs(

int <i>ta0</i> ,	TEXA register ta0
int <i>aem</i> ,	TEXA register aem
int <i>ta1</i>)	TEXA register ta1

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

sceHiGsCtxSetRegTexa (sceHiGsStdCtx, ta0, aem, ta1);

Return value

Table 5-106

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsXyoffsetRegs

Set XYOFFSET register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax**sceHiErr sceHiGsXyoffsetRegs(****int ofx,** XYOFFSET register ofx**int ofy)** XYOFFSET register ofy**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

sceHiGsCtxSetRegXyoffset (sceHiGsStdCtx, ofx, ofy);

Return value**Table 5-107**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

sceHiGsZbufRegs

Set ZBUF register

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.2	March 29, 2002

Syntax

sceHiErr sceHiGsZbufRegs(

int <i>zbp</i> ,	ZBUF register fbp
int <i>psm</i> ,	ZBUF register psm
int <i>zmsk</i>)	ZBUF register zmsk

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets a value in the standard port's register setting area.

This function is retained for compatibility with previous libraries.

It has been replaced by the following function:

sceHiGsCtxSetRegZbuf (sceHiGsStdCtx, zbp, psm, zmsk);

Return value

Table 5-108

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful

Chapter 6: Memory Area Management Services

Table of Contents

Functions	6-3
sceHiMemAlign	6-3
sceHiMemAlloc	6-4
sceHiMemCalloc	6-5
sceHiMemFree	6-6
sceHiMemGetNousedSize	6-7
sceHiMemGetUsedSize	6-8
sceHiMemInit	6-9
sceHiMemRealloc	6-10

Functions

sceHiMemAlign

Allocate memory with alignment

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
void* sceHiMemAlign(
    size_t bound;           Boundary size
    size_t size;)           Area Size
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function allocates a heap area of the specified size with the specified boundary and returns a pointer to that area.

The boundary size must always be a power of 2 (2^n).

Return value

Returns pointer to allocated area.

If allocation fails, NULL is returned.

sceHiMemAlloc

Allocate memory

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
void* sceHiMemAlloc(  
    size_t size;)           Area Size
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function allocates a heap area of the specified size and returns a pointer to the allocated area.

The size is specified in terms of bytes.

Return value

Returns pointer to allocated area.

If allocation fails, NULL is returned.

sceHiMemCalloc

Allocate memory with initialization

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
void* sceHiMemCalloc(
    size_t n;                Number
    size_t size;)            Area Size
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function allocates n contiguous sections of the specified area size and returns a pointer to the allocated area.

The area is completely initialized to zero.

Return value

Returns pointer to allocated area.

If allocation fails, NULL is returned.

sceHiMemFree

Free memory

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
void sceHiMemFree(  
    void *p;)           Pointer
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function frees the allocated heap area.

No check is performed to determine whether or not the specified address is a valid address.

Return value

None

sceHiMemGetUsedSize

Get used heap size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

sceHiErr sceHiMemGetUsedSize(

```
int *size)
```

Address of variable for receiving size of heap being used

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns in the *size* argument the size of the heap managed by the library that is being used, in bytes.

Return value

Table 6-2

Value	Macro	Result
0	SCE_HIG_NO_ERR	Success

sceHiMemRealloc

Reallocate memory

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhig	2.1	March 29, 2002

Syntax

```
void* sceHiMemRealloc(
    void *ptr;           Pointer
    size_t size;)        Area size
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function reallocates the allocated area according to the specified size and returns a pointer to the allocated area.

Return value

Returns pointer to allocated area.

If allocation fails, NULL is returned.

Chapter 7: High Level Graphics Plugin Library

Table of Contents

Structures	7-5
sceHiPlugAnimeData_t	7-5
sceHiPlugAnimeHead_t	7-6
sceHiPlugAnimePreCalcArg_t	7-8
sceHiPlugCameraData_t	7-9
sceHiPlugClipData_t	7-10
sceHiPlugClipHead_t	7-11
sceHiPlugClutBumpData_t	7-12
sceHiPlugClutBumpHead_t	7-13
sceHiPlugClutBumpPreArg_t	7-14
sceHiPlugFishEyeInitArg_t	7-15
sceHiPlugFishEyePreArg_t	7-16
sceHiPlugHrchyData_t	7-17
sceHiPlugHrchyHead_t	7-18
sceHiPlugHrchyMatrix_t	7-19
sceHiPlugHrchyPreCalcArg_t	7-20
sceHiPlugLightMapInitArg_t	7-21
sceHiPlugMicroAttr_t	7-22
sceHiPlugMicroData_t	7-23
sceHiPlugMicroInitArg_t	7-24
sceHiPlugMicroLight_t	7-25
sceHiPlugMicroPreCalcArg_t	7-26
sceHiPlugMicroTbl_t	7-27
sceHiPlugReflectPreArg_t	7-28
sceHiPlugRefractPreArg_t	7-29
sceHiPlugShadowBoxData_t	7-30
sceHiPlugShadowMapInitArg_t	7-31
sceHiPlugShapeHead_t	7-32
sceHiPlugShapeMatrix_t	7-34
sceHiPlugShareData_t	7-35
sceHiPlugShareHead_t	7-36
sceHiPlugSkinData_t	7-37
sceHiPlugSkinHead_t	7-38
sceHiPlugTex2DData_t	7-39
sceHiPlugTex2DHead_t	7-40
sceHiPlugTex2dInitArg_t	7-41
sceHiPlugTim2Data_t	7-42
sceHiPlugTim2Head_t	7-43
sceHiPlugTim2InitArg_t	7-44
Functions	7-45
FRAME_PLUG	7-45
sceHiPlugAnime	7-46
sceHiPlugAnimeGetData	7-48
sceHiPlugAnimeGetFrame	7-49
sceHiPlugAnimeGetHead	7-50
sceHiPlugAnimeGetKeyHead	7-51
sceHiPlugAnimeGetValue	7-52

sceHiPlugCamera	7-53
sceHiPlugCameraGetData	7-55
sceHiPlugClip	7-56
sceHiPlugClipGetData	7-58
sceHiPlugClipGetHead	7-59
sceHiPlugClutBump	7-60
sceHiPlugClutBumpGetData	7-62
sceHiPlugClutBumpGetHead	7-63
sceHiPlugClutBumpGetNormal	7-64
sceHiPlugFishEye	7-65
sceHiPlugHrchy	7-67
sceHiPlugHrchyGetData	7-69
sceHiPlugHrchyGetHead	7-70
sceHiPlugHrchyGetMatrix	7-71
sceHiPlugHrchyGetPivot	7-72
sceHiPlugLightMap	7-73
sceHiPlugManime	7-75
sceHiPlugMatrix	7-77
sceHiPlugMicro	7-79
sceHiPlugMicroGetData	7-81
sceHiPlugReflect	7-82
sceHiPlugRefract	7-84
sceHiPlugShadowBox	7-86
sceHiPlugShadowBoxGetData	7-88
sceHiPlugShadowMap	7-89
sceHiPlugShape	7-91
sceHiPlugShapeGetDataHead	7-94
sceHiPlugShapeGetGeometryColor	7-95
sceHiPlugShapeGetGeometryHead	7-96
sceHiPlugShapeGetGeometryNormal	7-97
sceHiPlugShapeGetGeometryST	7-98
sceHiPlugShapeGetGeometryVertex	7-99
sceHiPlugShapeGetHead	7-100
sceHiPlugShapeGetMaterialAttrib	7-101
sceHiPlugShapeGetMaterialGiftag	7-102
sceHiPlugShapeGetMaterialHead	7-103
sceHiPlugShapeGetMatrix	7-104
sceHiPlugShapeInvisible	7-105
sceHiPlugShapeMasterChainSetting	7-106
sceHiPlugShare	7-107
sceHiPlugShareGetData	7-109
sceHiPlugShareGetDst	7-110
sceHiPlugShareGetHead	7-111
sceHiPlugShareGetIndex	7-112
sceHiPlugShareGetShare	7-113
sceHiPlugShareGetSrc	7-114
sceHiPlugSkin	7-115
sceHiPlugSkinGetBW	7-118
sceHiPlugSkinGetData	7-119
sceHiPlugSkinGetHead	7-120
sceHiPlugSkinGetLB	7-121

sceHiPlugSkinGetLW	7-122
sceHiPlugTex2D	7-123
sceHiPlugTex2DGetClut	7-125
sceHiPlugTex2DGetData	7-126
sceHiPlugTex2DGetEnv	7-127
sceHiPlugTex2DGetHead	7-128
sceHiPlugTex2DGetTexel	7-129
sceHiPlugTex2DSize	7-130
sceHiPlugTim2	7-131
sceHiPlugTim2GetData	7-133
sceHiPlugTim2GetHead	7-134
sceHiPlugTim2GetName	7-135
sceHiPlugTim2GetNPictures	7-136
sceHiPlugTim2Num	7-137
sceHiPlugTim2SetData	7-138
sceHiPlugTim2SetPicture	7-139

Structures

sceHiPlugAnimeData_t

Anime data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef struct {  
    int hrchy;           Hrchy data ID  
    int numframe;        Number of frames  
    int keyframe;        KeyFrame data ID  
    int keyvalue;        KeyValue data ID  
} sceHiPlugAnimeData_t;
```

Description

This structure is returned by the sceHiPlugAnimeGetData() data access function.

It represents one Anime data structure of an ANIME_DATA data block.

sceHiPlugAnimeHead_t

Anime header union

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	December 6, 2002

Structure

```
typedef union {
    struct {
        u_int type;           Complementary type or function curve specification
        int index;           KeyFrame data or KeyValue data index
        size_t size;         KeyFrame data or KeyValue data word size
        int num;             Number of KeyFrame data or KeyValue data items
    } key;                  Structure indicating one KeyFrame data or KeyValue data item
    struct {
        int reserve[3];
        int num;             Number of data items
    } top;                  Structure indicating first header of data block
} sceHiPlugAnimeHead_t;
```

Description

This structure is returned by the sceHiPlugAnimeGetHead() and sceHiPlugAnimeGetKeyHead() data access functions.

The sceHiPlugAnimeGetHead() function uses the top structure for initial header information of the ANIME_DATA, KEYFRAME, and KEYVALUE data blocks.

The sceHiPlugAnimeGetKeyHead() function uses the key structure for header information of a single data item in KEYFRAME or KEYVALUE.

Interpolation types are expressed by bit flags as follows.

Table 7-1

Interpolation method	Interpolation type value
Constant (CONSTANT)	(1<<0)
Linear interpolation (LINEAR)	(1<<1)
Hermite interpolation (HERMITE)	(1<<2)
Bezier interpolation (BEZIER)	(1<<3)
B-SPLINE interpolation (BSPLINE) *	(1<<4)

Function curves are expressed using the following bit flag logical OR.

Table 7-2

Function curve	Curve value
TXYZ	$(1 \ll 10)$
RXYZ	$(1 \ll 11)$
RXZY	$(1 \ll 12)$
RYXZ	$(1 \ll 13)$
RYZX	$(1 \ll 14)$
RZXY	$(1 \ll 15)$
RZYG	$(1 \ll 16)$
SZYX	$(1 \ll 17)$
MATRIX	$(1 \ll 18)$

sceHiPlugAnimePreCalcArg_t

Anime plugin argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.2	July 2, 2001

Structure

```
typedef struct {
    u_int setframe;           Frame number passed to Anime plugin
    int setframe_enable;      1: Pass setframe to plugin
                             0: Do not pass
    u_int currentframe;       Current frame number returned by Anime plugin
} sceHiPlugAnimePreCalcArg_t;
```

Description

This is an argument type passed to the SCE_HIG_PRE_PROCESS of the Anime plugin. Set the address of the variable of this type to the arg member of a SceHiPlug type variable to allow it to be passed as an argument.

sceHiPlugCameraData_t

Camera data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Structure

```
typedef struct {
    int reserve[4];
    sceVu0FVECTOR screen;           Width, height, distance, clip additional size
    sceVu0FVECTOR window;           Aspect x,y, center x,y
    sceVu0FVECTOR depth;           Minimum, maximum, near, far
    sceVu0FVECTOR position;         Viewpoint
    sceVu0FVECTOR rotation;         Rotation
    sceVu0FVECTOR interest;         Point of interest
    sceVu0FVECTOR upvector;         Up vector
}sceHiPlugCameraData_t;
```

Description

This structure id returned by the data access function sceHiPlugCameraGetData().

It represents a CAMERA_DATA data block.

sceHiPlugClipData_t

Clip data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5	May 10, 2002

Structure

```
typedef struct {
    sceVu0FVECTOR min;           Minimum bounding box size
    sceVu0FVECTOR max;          Maximum bounding box size
}sceHiPlugClipData_t;
```

Description

This structure is returned by the data access function sceHiPlugClipGetData().
It represents one Clip data structure of a CLIP_DATA data block.

sceHiPlugClipHead_t

Clip header structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5	May 10, 2002

Structure

```
typedef struct {  
    int reserve[3];  
    int num;                Number of Clip data  
}sceHiPlugClipHead_t;
```

Description

This structure is returned by the data access function sceHiPlugClipGetHead().

It represents initial header information of a CLIP_DATA data block.

sceHiPlugClutBumpData_t

ClutBump data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```

typedef struct {
    int shape;           Shape data ID
    int tex2d;           Tex2D data ID
    int normal;          Normal table data ID
    int reserve;
} sceHiPlugClutBumpData_t;
    
```

Description

This structure is returned by the sceHiPlugClutBumpGetData() data access function.

It represents one ClutBump data structure of a CLUTBUMP_DATA data block.

sceHiPlugClutBumpHead_t

ClutBump header structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef struct {  
    int reserve[3];  
    int num;                                     Number of ClutBump data or Normal table data items  
} sceHiPlugClutBumpHead_t;
```

Description

This structure is returned by the sceHiPlugClutBumpGetHead() data access function.

It represents initial header information of a CLUTBUMP_DATA or CLUTBUMP_NORMAL data block.

sceHiPlugClutBumpPreArg_t

ClutBump plug-in argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	July 2, 2001

Structure

```
typedef struct {
    sceVu0FVECTOR light_dir;           Light direction
    sceVu0FVECTOR shading;             Shading parameter
} sceHiPlugClutBumpPreArg_t;
```

Description

The light vector for shading calculations is set in *light_dir*.

The following elements are entered in *shading*.

- shading[0] ambient alpha value
- shading[1] diffuse alpha value
- shading[2] specular alpha value
- shading[3] shininess value

The CLUT alpha is calculated from these parameters by using the following expression. (.) indicates the vector inner product.

CLUT alpha = ambient alpha + diffuse alpha * (light.normal) + specular alpha *(light.normal)^shininess

sceHiPlugFishEyeInitArg_t

FishEye plug-in argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	July 2, 2001

Structure

```
typedef struct {
    u_int zdepth;           Z-buffer storage format
    float rmin;             Minimum radius when performing Z-buffering
    float rmax;             Maximum radius when performing Z-buffering
} sceHiPlugFishEyeInitArg_t;
```

Description

Specify SCE_GS_PSMZ32, SCE_GS_PSMZ24, SCE_GS_PSM16, or SCE_GS_PSM16S for *zdepth*.

rmin and *rmax* set the effective range when Z-buffering is performed.

If the distance between the camera position and vertex in the viewpoint coordinate system is within this range, Z-buffering will be performed properly.

sceHiPlugFishEyePreArg_t

FishEye plug-in argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	July 2, 2001

Structure

```
typedef struct {
    sceVu0FVECTOR *camera_pos;           Pointer to viewpoint position
    sceVu0FVECTOR *camera_zdir;          Pointer to line-of-sight direction
    sceVu0FVECTOR *camera_up;            Pointer to vertical direction
    float tex_size;                       Rendering size
} sceHiPlugFishEyePreArg_t;
```

Description

The world view matrix is created from *camera_pos*, *camera_zdir*, and *camera_up*.
For *tex_size*, you can specify a special spherical size for a fish eye lens effect, which differs from the actual rendering rectangle size. However, the SCISSOR register must be set correctly.

sceHiPlugHrchyData_t

Hrchy data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef struct {
    sceVu0FVECTOR trans;           Translation
    sceVu0FVECTOR rot;             Rotation
    sceVu0FVECTOR scale;          Scaling
    int shape;                     Shape data ID
    int parent;                   Parent hierarchy ID
    int child;                   Child hierarchy ID (currently unused)
    int sibling;                 Sibling hierarchy ID (currently unused)
} sceHiPlugHrchyData_t;
```

Description

This structure is returned by the sceHiPlugHrchyGetData() data access function.

It represents one hierarchy structure of a HRCHY_DATA data block.

sceHiPlugHrchyHead_t

Hrchy header structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```

typedef struct {
    int reserve[2];
    u_int rorder;           Rotation order specification (unused for PIVOT)
    int num;                Number of HRCHY data or PIVOT data items
} sceHiPlugHrchyHead_t;

```

Description

This structure is returned by the sceHiPlugHrchyGetHead() data access function.

It represents initial header information of a HRCHY_DATA or PIVOT_DATA data block.

sceHiPlugHrchyMatrix_t

Hierarchy matrix data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.7	December 6, 2002

Structure

```
typedef struct {
    int reserve[3];
    int parent;                Parent layer ID
    sceVu0FMATRIX matrix;     Model local matrix
}sceHiPlugHrchyMatrix_t;
```

Description

This structure is returned by the sceHiPlugHrchyGetMatrix() data access function.

It represents a one-layer matrix structure of a MATRIXDATA data block.

sceHiPlugHrchyPreCalcArg_t

Hrchy plugin argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4	October 11, 2001

Structure

```
typedef struct {
    sceVu0FMATRIX *root;           Pointer to matrix to be multiplied by root hierarchy
} sceHiPlugHrchyPreCalcArg_t;
```

Description

By setting the address of a variable of this type for the arg member of a variable of type sceHiPlug, which is the argument type to be passed to the Hrchy plugin SCE_HIG_PRE_PROCESS, this structure can be passed as an argument.

By multiplying an arbitrary matrix by the root hierarchy, which is the highest level in the hierarchy structure, the coordinate transformation of a hierarchy under the root can be controlled.

sceHiPlugLightMapInitArg_t

LightMap plug-in argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	July 2, 2001

Structure

```
typedef struct {
    int width;           LightMap texture width
    int height;          LightMap texture height
    int fov;             Field of view criterion; specify TRUE for width, and
                        FALSE for height
} sceHiPlugLightMapInitArg_t;
```

Description

width and *height* specify the size of the texture to be used as the light map.

fov specifies whether the width or height is to be used as the field of view criterion for generating the texture matrix. When *fov*=TRUE, the width is used for the criterion, and When *fov*=FALSE, the height is used. If the texture is a square, either specification is OK.

sceHiPlugMicroAttr_t

Micro attribute constant

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.2	March 26, 2001

Structure

```
typedef enum {
    SCE_HIP_MICRO_ATTR_NONE,
    SCE_HIP_MICRO_ATTR_FGE,
    SCE_HIP_MICRO_ATTR_ANTI
} sceHiPlugMicroAttr_t;
```

Description

The sceHiPlugMicroAttr_t type is a constant for specifying attributes of a microprogram. Each attribute value is handled as a bit value and can be duplicated and held. Use "bit or" in a microprogram attribute specification that has multiple attributes.

Example: Microprogram attribute that handles Fog and performs Anti.

```
SCE_HIP_MICRO_ATTR_FGE | SCE_HIP_MICRO_ATTR_ANTI
```

Table 7-3

Constant	Meaning
SCE_HIP_MICRO_ATTR_NONE	No special attribute
SCE_HIP_MICRO_ATTR_FGE	Fog attribute
SCE_HIP_MICRO_ATTR_ANTI	Anti attribute

sceHiPlugMicroData_t

Micro data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef struct {
    sceVu0FMATRIX wscreen;           World screen matrix
    union {
        sceVu0FMATRIX wview;        World view matrix
        sceVu0FMATRIX texproj;      Texture projection matrix
    } mtx;                           Structure indicating a special matrix area
    sceVu0FMATRIX material;          Material data
                                     This area is overwritten by the Shape data material data
    float camx,camy,camz;            Camera position
    float aa1;                       AA1 cutoff value
    float fogA,fogB;                 Fog parameter
    u_int prmode;                    Gs PRIMODE register bit
    int reserve;
    union {
        sceVu0FVECTOR clip;          Clip parameter
        struct {
            float texsize;            Spherical texture size
            float ZA,ZB;              Depth parameters
            int reserve;
        } fisheye;                  Fisheye lens parameter structure
    } clip;                          Structure indicating clip or fisheye lens parameter
    float shift;                     EmbossBump texture shift amount
    float refidx;                    Refraction rate
    float zoom;                      Reflection/Refraction texture zoom size
    float zshift;                    Reflection/Refraction Z shift amount
    sceHiPlugMicroLight_t light[3];  Parallel light source, point light, or spotlight information
} sceHiPlugMicroData_t;
```

Description

This is the MICRO_DATA data block structure.

The required parameters differ depending on the corresponding microprogram.

The material data shown here is overwritten by the material data in the SHAPE_DATA data block. Member names are defined here for convenience.

This data block is transferred directly to VU1 local memory according to the Micro plug-in.

sceHiPlugMicroInitArg_t

Micro plugin argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.2	July 2, 2001

Structure

```

typedef struct {
    sceHiPlugMicroTbl_t *tbl;           Pointer to table that defines addresses and attributes
                                       of microprogram
    u_int tblnum;                       Number saved in tbl
} sceHiPlugMicroInitArg_t;

```

Description

This is an argument type that is accepted in the SCE_HIG_INIT_PROCESS of a Micro plugin.

Set the address of the variable of this type to the *arg* member of a SceHiPlug type variable to allow it to be passed as an argument.

When this structure is not passed to a Micro plugin, the default microprogram is selected.

sceHiPlugMicroLight_t

MicroLight structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	December 6, 2002

Structure

```
typedef struct {  
    sceVu0FVECTOR dir[4];           Light direction  
    sceVu0FVECTOR pos[4];           Light position  
    sceVu0FVECTOR col[4];           Light color + 1 ambient color  
} sceHiPlugMicroLight_t;
```

Description

This structure represents light information in a MICRO_DATA data block.

The fourth light color of the parallel light source is the ambient color for all lights.

Note that all light directions are minus transposed.

sceHiPlugMicroPreCalcArg_t

Micro plugin argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.2	July 2, 2001

Structure

```
typedef struct {
    int micro;                Microprogram number to be transferred (used).
                              Number passed with the SCE_HIG_INIT_PROCESS
                              that is assumed to be 0 at the head of the table

    float anticutoff;         Anti parameter passed to microprogram

    float fogbegin;           Fog parameter passed to microprogram

    float fogend;             Fog parameter passed to microprogram
} sceHiPlugMicroPreCalcArg_t;
```

Description

Argument type passed to the SCE_HIG_PRE_PROCESS of the microprogram. Set the address of the variable of this type to the arg member of a sceHiPlug type variable to allow it to be passed as an argument.

The microprogram is not transferred when -1 is given to the micro member. When using the *anticutoff*, *fogbegin* and *fogend* members, the corresponding microprograms must be registered.

sceHiPlugMicroTbl_t

Microprogram registration table type

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.2	July 2, 2001

Structure

```
typedef struct {  
    u_int *micro;           Microprogram pointer  
    u_int attr;             Microprogram attribute  
} sceHiPlugMicroTbl_t
```

Description

This is a table type for registering microprograms. The member *tbl* of argument type `sceHiPlugMicroInitArg_t` that micro plugins accept must be this type of array.

attr must be an OR constant defined in `sceHiPlugMicroAttr_t`.

sceHiPlugReflectPreArg_t

Reflect plug-in argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	October 11, 2001

Structure

```
typedef struct {
    sceVu0FVECTOR *camera_pos;           Pointer to viewpoint position
    sceVu0FVECTOR *camera_zdir;          Pointer to line-of-sight direction
    sceVu0FVECTOR *camera_up;            Pointer to vertical direction
    float zoom;                           Zoom percentage
    float z_shift;                         Amount of Z-shift
} sceHiPlugReflectPreArg_t;
```

Description

The world view matrix is created from *camera_pos*, *camera_zdir* and *camera_up*.

zoom and *z_shift* are used to calculate a suitable ST vector for reflection.

The following expressions are used for calculating the ST vector. (.) indicates the vector inner product.

```
reflect=2*(normal,eye)-eye
reflect.x=reflect.x*zoom
reflect.y=reflect.y*zoom
m=sqrt(reflect.x^2+reflect.y^2+(reflect.z+z_shift)^2)
S= reflect.x/(2*m)+0.5
T= reflect.y/(2*m)+0.5
```


sceHiPlugRefractPreArg_t

Refract plug-in argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	October 11, 2001

Structure

```
typedef struct {
    sceVu0FVECTOR *camera_pos;           Pointer to viewpoint position
    sceVu0FVECTOR *camera_zdir;          Pointer to line-of-sight direction
    sceVu0FVECTOR *camera_up;            Pointer to vertical direction
    float refract_index;                  Refraction percentage
    float zoom;                           Zoom percentage
    float z_shift;                         Amount of Z-shift
} sceHiPlugRefractPreArg_t;
```

Description

The world view matrix is created from *camera_pos*, *camera_zdir* and *camera_up*.

zoom and *z_shift* are used to calculate a suitable ST vector for reflection.

The following expressions are used for calculating the ST vector. (.) indicates the vector inner product.

```
refract=2*(refract_index*normal.eye)+eye
refract.x=refract.x*zoom
refract.y=refract.y*zoom
m=sqrt(refract.x^2+refract.y^2+(refract.z+z_shift)^2)
S= refract.x/(2*m)+0.5
T= refract.y/(2*m)+0.5
```

sceHiPlugShadowBoxData_t

ShadowBox data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```

typedef struct {
    sceVu0FVECTOR min;           Bounding box minimum size
    sceVu0FVECTOR max;           Bounding box maximum size
    sceVu0FVECTOR box[8];        Bounding box size after operation is performed
} sceHiPlugShadowBoxData_t;

```

Description

This structure is returned by the sceHiPlugShadowBoxGetData() data access function.

It represents information for a SHADOWBOX_DATA data block.

sceHiPlugShadowMapInitArg_t

ShadowMap plug-in argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	July 2, 2001

Structure

```
typedef struct {
    int width;           Shadow texture width
    int height;          Shadow texture height
    u_int *box;          Pointer to ShadowBox
} sceHiPlugShadowMapInitArg_t;
```

Description

width and *height* specify the shadow texture size.

box is the bounding box to be used for generating the shadow matrix.

NULL can be specified if the ShadowMap plug-in block is maintaining the ShadowBox data block (SCE_HIP_SHADOWBOX_DATA).

ShadowBox is the bounding box that is generated using a diagonal line from the shadow texture rendering source and the maximum and minimum sizes of the shadow object.

To use the ShadowMap plug-in, you must provide this structure.

sceHiPlugShapeHead_t

Shape header union

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```

typedef union {
    struct {
        int id;                Geometry data ID
        size_t size;           Geometry data word size
        u_int prim;            Gs PRIM register bit
        int num;               Number of primitives
    } geo;                     Geometry data header structure
    struct {
        int id;                Material data ID
        int num;               Number of Geometry data items within Material data
        int tex_id;            Texture data ID within Material data
        int tex_num;           Number of Texture data items within Material data
    } mat;                     Material data header structure
    struct {
        int id;                Shape data ID
        size_t size;           Shape data word size
        int reserve;
        int num;               Number of Material data items within Shape data
    } dat;                     Shape data header structure
    struct {
        int reserve[3];
        int num;               Number of Shape data items within SHAPE_DATA or
                                number of Matrix items within BASEMATRIX
    } top;                     Structure indicating first header information of
                                SHAPE_DATA or structure indicating first header
                                information of BASEMATRIX
} sceHiPlugShapeHead_t;

```

Description

This is a union returned by the `sceHiPlugShapeGetHead()`, `sceHiPlugShapeGetDataHead()`, `sceHiPlugShapeGetMaterialHead()`, and `sceHiPlugShapeGetGeometryHead()` data access functions.

It represents header information of data blocks having SHAPE_DATA and BASEMATRIX data types, which belong to the SHAPE category.

The `sceHiPlugShapeGetHead()` function uses the top structure for initial header information of SHAPE_DATA and BASEMATRIX data blocks.

The `sceHiPlugShapeGetDataHead()` function uses the dat structure for Shape data header information of a single SHAPE_DATA item.

The `sceHiPlugShapeGetMaterialHead()` function uses the mat structure for Material data header information of a single SHAPE_DATA item.

The `sceHiPlugShapeGetGeometryHead()` function uses the `geo` structure for Geometry data header information of a single `SHAPE_DATA` item.

sceHiPlugShapeMatrix_t

BASEMATRIX data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```

typedef struct {
    int reserve[2];
    int flags;                Shape data visible/invisible flag
    int shape;                Shape data ID
    sceVu0FMATRIX local;     Local world matrix
    sceVu0FMATRIX light;     Light rotation matrix
} sceHiPlugShapeMatrix_t;

```

Description

This structure is returned by the sceHiPlugShapeGetMatrix() data access function.
It represents a single matrix structure of a BASEMATRIX data block.

sceHiPlugShareData_t

Share data union

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef union {
    struct {
        int offset;           SHAREVERTEX or SHARENORMAL offset position
        int geomid;          Geometry data ID
        int reserve;
        int num;              Number of SHAREVERTEX or SHARENORMAL data items
    } shr;                    Structure indicating SHAREVERTEX and SHARENORMAL
    struct {
        int voffset;          VERTEXINDEX offset position
        int vlength;          Vertex length (number) from VERTEXINDEX offset position
        int noffset;          NORMALINDEX offset position
        int nlength;          Normal length (number) from NORMALINDEX offset position
    } dat;                    Structure indicating SHARE_DATA
} sceHiPlugShareData_t;
```

Description

This structure is returned by the sceHiPlugShareGetData() and sceHiPlugShareGetShare() data access functions.

The sceHiPlugShareGetData() function uses the **dat** structure for SHARE_DATA data.

The sceHiPlugShareGetShare() function uses the **shr** structure for SHAREVERTEX and SHARENORMAL data.

sceHiPlugShareHead_t

Share header union

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```

typedef struct {
    int reserve[2];
    int shape;                Shape data ID; ued only for SHAREVERTEX and
                              SHARENORMAL
    int num;                  Number of data items
} sceHiPlugShareHead_t;
  
```

Description

This structure is returned by the sceHiPlugShareGetHead() data access function.

It represents initial header information of SHARE_DATA, VERTEXINDEX, NORMALINDEX, SRCDSTVERTEX, SRCDSTNORMAL, SHAREVERTEX, and SHARENORMAL data blocks.

sceHiPlugSkinData_t

Skin data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Structure

```
typedef struct {
    sceVu0FVECTOR vertex;      Vertex data
    sceVu0FVECTOR normal;     Normal data
    sceVu0FVECTOR weight;     Vertex weight coefficients (max 4)
    int matrix[4];
    int id[4];                 Bone ID in BASEMATRIX corresponding to weight coefficients
}sceHiPlugClipData_t;
```

Description

This structure is returned by the data access function sceHiPlugSkinGetData().
It represents one Skin data structure of a SKIN_DATA data block.

sceHiPlugSkinHead_t

Skin header structure

Library	Introduced	Documentation last modified
libhip	2.5.3	July 8, 2002

Structure

```
typedef struct {
    int reserve[2];
    int func;           Function option
    int num;           Skin data count
}sceHiPlugSkinHead_t;
```

Description

This structure is returned by the data access function sceHiPlugSkinGetHead(). It represents the initial header information of each of the data blocks SKIN_DATA, SKIN_LB, SKIN_LW, and SKIN_BW. The member *func* is used only for the SKIN_DATA data block.

- Low-order 16 bits of func
 - Options for processing the normals of skin
 - 0: Perform normalization when calculating normals
 - 1: Do not perform normalization when calculating normals
 - 2: Reserved
 - 3: Do not calculate normals
- High-order 16 bits of func
 - BASEMATRIX shape options for parent shape to which skin is attached
 - 0: Include only rotation and translation, and perform associated processing for the matrix for which the fourth row is {0,0,0,1}
 - 1: Perform associated processing for a matrix that can be inverted and for which the fourth row is {0,0,0,1}
 - 2: Perform associated processing only for the unit matrix

sceHiPlugTex2DData_t

Tex2D data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef struct {
    sceGsTex0 tex0;           Gs TEX0 register
    u_long addr;              Gs TEX0 register address
    size_t texelsize;         Texel word size
    size_t clutsize;          CLUT word size
    u_short texelwidth;       Texel width
    u_short texelheight;     Texel height
    u_short clutwidth;        CLUT width
    u_short clutheight;      CLUT height
} sceHiPlugTex2DData_t;
```

Description

This structure is returned by the `sceHiPlugTex2DGetData()` data access function.

It represents one set of texture information for a data block.

sceHiPlugTex2DHead_t

Tex2D header structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef struct {  
    int reserve[3];  
    int num;                Number of TEX2D_DATA textures or number of  
                           TEX2D_ENV texture environments  
} sceHiPlugTex2DHead_t;
```

Description

This structure is returned by the sceHiPlugTex2DGetHead() data access function.

It represents initial header information of TEX2D_DATA and TEX2D_ENV data blocks.

sceHiPlugTex2dInitArg_t

Tex2D plugin argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.2	July 2, 2001

Structure

```
typedef struct {
    int resident;                Flag for switching between resident / non-resident
                                texture data GS Memory

    sceHiGsMemTbl *tbl;         GS Service structure for users to designate TBP/CBP
                                settings
} sceHiPlugTex2dInitArg_t;
```

Description

Argument type accepted by the SCE_HIP_INIT_PROCESS of the Tex2D plugin. Set the address of the variable of this type to the arg member of a SceHiPlug type variable to allow it to be passed as an argument.

When resident=TRUE and tbl=NULL, the texture size is allocated internally in the library. When SCE_HIG_END_PROCESS is called, the texture size is freed.

sceHiPlugTim2Data_t

Tim2 data structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```

typedef struct {
    int id;                Tim2 data ID
    int *ptr;              Pointer to loaded file (set by user)
    size_t size;           Word size of file to be loaded
    size_t length;         Length of filename; currently, fixed at 16 characters
    char fname[16];        Filename to be loaded; currently, fixed at up to 16
                           characters
} sceHiPlugTim2Data_t;

```

Description

This structure is returned by the sceHiPlugTim2GetData() data access function.

It represents one Tim2 data structure for a TIM2_DATA data block.

The parsing of the Tim2 format needs to be done as a separate operation.

sceHiPlugTim2Head_t

Tim2 header structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	January 4, 2002

Structure

```
typedef struct {
    int reserve[3];
    int num;                      Number of Tim2 data items
} sceHiPlugTim2Head_t;
```

Description

This structure is returned by the sceHiPlugTim2GetHead() data access function.

It represents initial header information of a TIM2_DATA data block.

The parsing of the Tim2 format needs to be done as a separate operation.

sceHiPlugTim2InitArg_t

Tim2 plugin argument structure

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.2	July 2, 2001

Structure

```
typedef struct {  
    int resident;                Flag for switching between resident / non-resident  
                                texture data GS Memory  
  
    sceHiGsMemTbl *tbl;          GS Service structure for users to designate TBP/CBP  
                                settings  
  
} sceHiPlugTim2dInitArg_t;
```

Description

Argument type accepted by the SCE_HIG_INIT_PROCESS of the Tim2 plugin. Set the address of the variable of this type to the arg member of a SceHiPlug type variable to allow it to be passed as an argument.

Functions

FRAME_PLUG

Frame plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.1	March 29, 2002

Syntax

(No plugin function)

(No plugin arguments)

Description

This is a virtual plugin that has no plugin function.

Inserted plugin functions are executed as a batch.

The dependency relationships of plugin calls is described by the order in the inserted plugin block list.

Inserted plugins are executed by calling sceHiCallPlug().

Basically, this virtual plugin can be inserted in any plugin block.

The user can customize the frame plugin block to call an appropriate plugin.

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes

Table 7-4

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_FRAME	FRAME_PLUG

Data Type Attributes

None

Return value

(No plugin return value)

sceHiPlugAnime

Anime plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.1	December 6, 2002

Syntax

sceHiErr sceHiPlugAnime(

```
sceHiPlug *plug;
```

Pointer to plugin block

```
int process);
```

Plugin process ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function interpolates values for translating, rotating, or zooming hierarchy data between keyframes.

The following processing is called according to the value specified by the process argument.

Table 7-5

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Reserves internal buffers. Reads hierarchy structure data, keyframe data, key value data and animation data.
SCE_HIG_PRE_PROCESS	Interpolates coordinate conversions, motion, rotation and expansion of hierarchy structure data between keyframes. The following process also occurs when an actual plugin block argument (HiPlug type member args) is present (not NULL). When element setframe_enable of an actual argument is 1, sets the value of element setframe as the current keyframe. Always returns the current keyframe number to element currentframe of the actual argument.
SCE_HIG_POST_PROCESS	Performs no processing.
SCE_HIG_END_PROCESS	Frees internal buffers.

The amount of memory consumed when this plugin is used is shown below.

Take this amount into account when using the `sceHiMemInit()` function in order to specify a size large enough for the internal buffer.

This size is consumed for each individual plugin block.

Amount of Memory Consumed**Table 7-6:**

Use	Amount Consumed
Internal buffer	$2 \text{ (word/keyframe)} + 1 \text{ (word/keyvalue)} + 3 \text{ (qword)}$

The type attributes of this plugin and the type attributes of the required data and write permissions are shown below.

Plugin Type Attributes**Table 7-7**

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_ANIME_PLUG

Data Type Attributes**Table 7-8:**

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_ANIME_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_KEYFRAME	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_KEYVALUE	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_HRCHY_DATA	r/w

Return value**Table 7-9**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
2	SCE_HIG_INVALID_VALUE	A NULL pointer was provided
3	SCE_HIG_INVALID_DATA	- Data types do not match - Number of animations is 0 or less - Number of keyframes is 0 or less - Number of key values is 0 or less - The stack area is NULL
4	SCE_HIG_FAILURE	Data block cannot be found

sceHiPlugAnimeGetData

Get Anime data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```

sceHiPlugAnimeData_t *sceHiPlugAnimeGetData(
  sceHiPlugAnimeHead_t *h,           Header pointer
  int idx);                          Index

```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Anime data item having an ANIME_DATA data type.

The ANIME_DATA header pointer obtained from sceHiPlugAnimeGetHead() is assigned to **h*.

idx is set to the desired position of the Anime data item to be obtained.

Return value

sceHiPlugAnimeData_t type Pointer to Anime data

sceHiPlugAnimeGetFrame

Get Anime Frame data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
int *sceHiPlugAnimeGetFrame(
    sceHiPlugAnimeHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Animation Frame data item having a KEYFRAME data type.

The KEYFRAME header pointer obtained from sceHiPlugAnimeGetKeyHead() is assigned to *h.

idx is set to the desired position of the Frame data item to be obtained.

Return value

int type Pointer to Animation Frame data

sceHiPlugAnimeGetHead

Get Anime header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugAnimeHead_t *sceHiPlugAnimeGetHead(
    sceHiPlug *p;           Pointer to plug-in block
    sceHiType t);           Data type
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having an ANIME_DATA, KEYFRAME, or KEYVALUE data type, which belongs to the ANIME category.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugAnimeHead_t type Header pointer

sceHiPlugAnimeGetKeyHead

Get Anime Key header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugAnimeHead_t *sceHiPlugAnimeGetKeyHead(
    sceHiPlugAnimeHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Animation Key header having a KEYFRAME or KEYVALUE data type.

The KEYFRAME or KEYVALUE header pointer obtained from sceHiPlugAnimeGetHead() is assigned to **h*.

idx is set to the desired position of the Key header to be obtained.

Return value

sceHiPlugAnimeHead_t type Animation Key header pointer

sceHiPlugAnimeGetValue

Get Anime Value data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugAnimeGetValue(
    sceHiPlugAnimeHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Animation Value data item having a KEYVALUE data type.

The KEYVALUE header pointer obtained from sceHiPlugAnimeGetKeyHead() is assigned to **h*.

idx is set to the desired position of the Value data item to be obtained.

Return value

sceVu0FVECTOR type Animation Value data pointer

sceHiPlugCamera

Camera plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	December 6, 2002

Syntax

```
sceHiErr sceHiPlugCamera(
    sceHiPlug *plug;           Pointer to plugin block
    int process);              Plugin process identifier
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

The Camera plugin finds the screen matrix, view matrix, and clip volume from the specified Camera data and copies the result into the Micro data.

Table 7-10

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Allocates internal buffers. Generates screen matrix. Finds clip volume. Copies result to Micro data
SCE_HIG_PRE_PROCESS	Generates view matrix. Finds view screen matrix. Copies result to Micro data. Finds camera position and copies result to Micro data.
SCE_HIG_POST_PROCESS	Does nothing.
SCE_HIG_END_PROCESS	Frees internal buffers.

Memory Consumed

Table 7-11

Use	Amount Consumed
Packet management area	None
DMA packet buffer	None
Internal buffer	1 qword

Plugin Type Attributes

Table 7-12

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_CAMERA	SCE_HIP_CAMERA_PLUG

Data Type Attributes

Table 7-13

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_CAMERA	SCE_HIP_CAMERA_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_DATA	r/w

Return value

The following detailed error states can be found from the sceHiErrStateType variable.

Table 7-14

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing succeeded
1	SCE_HIG_NO_HEAP	Internal memory allocation failed
3	SCE_HIG_INVALID_DATA	- Data type SCE_HIP_CAMERA_DATA was not found - Data type SCE_HIP_MICRO_DATA was not found - The stack area is NULL

sceHiPlugCameraGetData

Get Camera data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Syntax

```
sceHiPlugCameraData_t *sceHiPlugCameraGetData(  
sceHiPlug *p);
```

Pointer to plugin block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a data pointer to the CAMERA_DATA data type.

If there is no data, NULL is returned.

Return value

sceHiPlugCameraData_t type Pointer to camera data

sceHiPlugClip

Clip plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5	December 6, 2002

Syntax

```
sceHiErr sceHiPlugClip(
    sceHiPlug *plug;           Pointer to plugin block
    int process;               Plugin process identifier
);
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

The Clip plugin performs object clipping of the external volume from the Clip data's bounding box and the clipping volume in the Micro data. It also controls drawing transfers.

The matrix invisibility flag is set to ON for a clipping object and OFF for a non-clipping object. If an object's invisibility flag is ON it will not perform a DMA transfer which allows wasted DMA transfers to be controlled.

For more information on the setting of the visibility / invisibility flag, please see the description of the sceHiPlugShapeInvisible() function.

Table 7-15

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Perform clipping of matrix units.
SCE_HIG_POST_PROCESS	No operation.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-16

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-17

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_CLIP	SCE_HIP_CLIP_PLUG

Data Type Attribute**Table 7-18**

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_CLIP	SCE_HIP_CLIP_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_BASEMATRIX	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_DATA	r/o

Return value

The following detailed error states can be obtained from the variable sceHiErrStateType.

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- Data type SCE_HIP_CLIP_DATA cannot be found. - Data type SCE_HIP_BASEMATRIX cannot be found. - Data type SCE_HIP_MICRO_DATA cannot be found. - The stack area is NULL

sceHiPlugClipGetData

Get Clip data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5	May 10, 2002

Syntax

```

sceHiPlugClipData_t *sceHiPlugClipGetData(
  sceHiPlugClipHead_t *h,           Header pointer
  int idx);                          Index

```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets one Clip data item of data type CLIP_DATA.

The CLIP_DATA header pointer obtained from sceHiPlugClipGetHead() is assigned to *h.

idx is set to the desired position of the Clip data item to be obtained.

Return value

sceHiPlugClipData_t type Pointer to Clip data

sceHiPlugClipGetHead

Get Clip header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5	May 10, 2002

Syntax

```
sceHiPlugClipHead_t *sceHiPlugClipGetHead(
    sceHiPlug *p;           Pointer to plugin block
    sceHiType t);           Data type
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block of CLIP DATA with a data type belonging to category CLIP.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugClipHead_t type Header pointer

sceHiPlugClutBump

ClutBump plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	December 6, 2002

Syntax

```
sceHiErr sceHiPlugClutBump(
    sceHiPlug *plug;           Pointer to plug-in block
    int process;               Plug-in process identifier
);
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function calculates the shading intensity from 256 normal line tables and writes it as the texture CLUT alpha value. It produces a bump effect by varying the brightness as a function of alpha blending, with the texture as a base.

Calculation expression:

$\text{CLUT alpha} = \text{ambient alpha} + \text{diffuse alpha} * (\text{light.normal}) + \text{specular alpha} * (\text{light.normal})^{\text{shininess}}$

Table 7-19

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Calculate CLUT alpha and write it to texture.
SCE_HIG_POST_PROCESS	No operation.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-20

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword + 1 qword/clut

Plug-in Type Attribute

Table 7-21

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_BUMP	SCE_HIP_CLUTBUMP_PLUG

Data Type Attribute**Table 7-22**

Repository	Project	Category	Data ID	r/w
SCE_HIP_	SCE_HIP_	SCE_HIP_	SCE_HIP_CLUTBUMP_	r/o
COMMON	FRAMEWORK	BUMP	DATA	
SCE_HIP_	SCE_HIP_	SCE_HIP_	SCE_HIP_	r/o
COMMON	FRAMEWORK	BUMP	CLUTBUMP_NORMAL	
SCE_HIP_	SCE_HIP_	SCE_HIP_	SCE_HIP_	r/o
COMMON	FRAMEWORK	SHAPE	BASEMATRIX	
SCE_HIP_	SCE_HIP_	SCE_HIP_	SCE_HIP_	r/w
COMMON	FRAMEWORK	TEX2D	TEX2D_DATA	

Return value**Table 7-23**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- Data type SCE_HIP_CLUTBUMP_DATA cannot be found. - Data type SCE_HIP_TEX2_DATA cannot be found. - Data type SCE_HIP_BASEMATRIX cannot be found. - Data type SCE_HIP_CLUTBUMP_NORMAL cannot be found. - The stack area is NULL

sceHiPlugClutBumpGetData

Get ClutBump data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```

sceHiPlugClutBumpData_t *sceHiPlugClutBumpGetData(
  sceHiPlugClutBumpHead_t *h,           Header pointer
  int idx);                             Index

```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single ClutBump data item having a CLUTBUMP_DATA data type.

The CLUTBUMP_DATA header pointer obtained from sceHiPlugClutBumpGetHead() is assigned to **h*.

idx is set to the desired position of the ClutBump data item to be obtained.

Return value

sceHiPlugClutBumpData_t type ClutBump data pointer

sceHiPlugClutBumpGetHead

Get ClutBump header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugClutBumpHead_t *sceHiPlugClutBumpGetHead(
    sceHiPlug *p;           Pointer to plug-in block
    sceHiType t);           Data type
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having CLUTBUMP_DATA or CLUTBUMP_NORMAL data types, which belongs to the CLUTBUMP category.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugClutBumpHead_t type Header pointer

sceHiPlugClutBumpGetNormal

Get ClutBump normal table

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugClutBumpGetNormal(  
    sceHiPlugClutBumpHead_t *h,           Header pointer  
    int idx);                             Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single normal table having a CLUTBUMP_NORMAL data type.
The CLUTBUMP_NORMAL header pointer obtained from sceHiPlugClutBumpGetHead() is assigned to *h.
idx is set to the desired position of the normal table to be obtained.
One normal table has 256 normal lines.

Return value

sceVu0FVECTOR type CLUTBUMP normal table pointer

sceHiPlugFishEye

FishEye plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	December 6, 2002

Syntax

sceHiErr sceHiPlugFishEye(

sceHiPlug *plug; Pointer to plug-in block
int process;); Plug-in process identifier

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function performs scene rendering with a fish eye lens effect. To perform scene rendering, a separate microprogram and Micro plug-in are required. The Micro plug-in is inserted in the data format.

A suitable drawing environment is required for fish eye lens rendering. For specific information about using the fish eye plug-in, see the sample source code and data.

Table 7-24

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Generate view matrix.
SCE_HIG_POST_PROCESS	Perform rendering with inserted Micro plug-in.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-25

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-26

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_REFLECT	SCE_HIP_FISHEYE_PLUG
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_PLUG

Data Type Attribute**Table 7-27**

Repository	Project	Category	Data ID	r/w
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ MICRO	SCE_HIP_ MICRO_DATA	r/w

Return value**Table 7-28**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- Plugin type SCE_HIP_MICRO_PLUG cannot be found. - Data type SCE_HIP_MICRO_DATA cannot be found. - The stack area is NULL

sceHiPlugHrchy

Hrchy plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.1	December 6, 2002

Syntax

```
sceHiErr sceHiPlugHrchy(
    sceHiPlug *plug;           Pointer to hierarchy plugin block
    int process);              Plugin process ID
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function creates a hierarchy matrix internally.

There is no hierarchy depth limitation.

The following processing is called according to the value specified by the *process* argument.

Table 7-29:

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Allocates internal buffers. Reads hierarchy data and matrix data.
SCE_HIG_PRE_PROCESS	Transforms coordinates, generates an accumulation matrix from translation, rotation, or zooming values and hierarchies, and reflects it in the data.
SCE_HIG_POST_PROCESS	Performs no processing.
SCE_HIG_END_PROCESS	Frees internal buffers.

The amount of memory consumed when this plugin is used is shown below.

Take this amount into account when using the `sceHiMemInit()` function in order to specify a size large enough for the internal buffer.

This size is consumed for each individual plugin block.

Amount of Memory Consumed

Table 7-30

Use	Amount Consumed
Internal buffer	1 (qword)

The type attributes of this plugin and the type attributes of the required data and write permissions are shown below.

Plugin Type Attributes**Table 7-31**

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_HRCHY_PLUG

Data Type Attributes**Table 7-32**

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_HRCHY_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_PIVOT_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_BASEMATRIX	r/w

Return value**Table 7-33**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
2	SCE_HIG_INVALID_VALUE	A NULL pointer was provided
3	SCE_HIG_INVALID_DATA	- Data types do not match - The stack area is NULL
4	SCE_HIG_FAILURE	Data block cannot be found

sceHiPlugHrchyGetData

Get HRCHY data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugHrchyData_t *sceHiPlugHrchyGetData(
    sceHiPlugHrchyHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single hierarchy structure having a HRCHY_DATA data type.

The HRCHY_DATA header pointer obtained from sceHiPlugHrchyGetHead() is assigned to *h.

idx is set to the desired position of the HRCHY data to be obtained.

Return value

sceHiPlugHrchyData_t type HRCHY data pointer

sceHiPlugHrchyGetHead

Get HRCHY header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```

sceHiPlugHrchyHead_t *sceHiPlugHrchyGetHead(
  sceHiPlug *p;                                Pointer to plug-in block
  sceHiType t);                                Data type

```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having a HRCHY_DATA or PIVOT_DATA data type, which belongs to the HRCHY category.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugHrchyHead_t type Header pointer

sceHiPlugHrchyGetMatrix

Get hierarchy matrix data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.7	December 6, 2002

Syntax

```
sceHiPlugHrchyData_t *sceHiPlugHrchyGetMatrix(
    sceHiPlugHrchyHead_t *h,           Header pointer
    int idx);                          Index
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets one-layer matrix data having a data type of MATRIXDATA.

The MATRIXDATA header pointer that was obtained from sceHiPlugHrchyGetHead() is assigned to *h.

idx specifies the number of the layer for which matrix data is to be obtained.

Return Value

sceHiPlugHrchyMatrix_t type Pointer to hierarchy matrix data

sceHiPlugHrchyGetPivot

Get PIVOT data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugHrchyGetPivot(  
    sceHiPlugHrchyHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single PIVOT data item having a PIVOT_DATA data type.
The PIVOT_DATA header pointer obtained from sceHiPlugHrchyGetHead() is assigned to *h.
idx is set to the desired position of the PIVOT data to be obtained.

Return value

sceVu0FVECTOR type PIVOT data pointer

sceHiPlugLightMap

Light map plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	December 6, 2002

Syntax

sceHiErr sceHiPlugLightMap(

sceHiPlug *plug; Pointer to plug-in block
int process;); Plug-in process identifier

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function produces a lighting effect by generating a texture projection matrix from light information that is in the micro data and projecting the light texture on an object.

A separate microcode and micro plug-in are required.

This function supports only parallel light source No. 0 within the light information that is in the micro data.

For SCE_HIG_INIT_PROCESS, sceHiPlugLightMapInitArg_t must be specified.

For specific information about using the light map plug-in, see the sample source code and data.

Table 7-34

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Generate texture projection matrix.
SCE_HIG_POST_PROCESS	Perform light mapping according to inserted micro plug-in.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-35

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attributes

Table 7-36

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_LIGHT	SCE_HIP_LIGHTMAP_PLUG
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_PLUG

Data Type Attribute

Table 7-37

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_DATA	r/w

Return value

Table 7-38

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
2	SCE_HIG_INVALID_VALUE	plug->args is NULL
3	SCE_HIG_INVALID_DATA	- Plugin type SCE_HIP_MICRO_PLUG cannot be found. - Data type SCE_HIP_MICRO_DATA cannot be found. - The stack area is NULL

sceHiPlugManime

Matrix animation plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.7	December 6, 2002

Syntax

```
sceHiErr sceHiPlugManime(
    sceHiPlug *plug;           Pointer to plug-in block
    int process;               Plug-in process identifier
);
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This plug-in interpolates between matrices that were assigned as keys.

The matrix is temporarily decomposed into translation and rotation components, and the rotation component is converted to a quaternion.

Each component is linearly interpolated to generate the result matrix. The generated matrix is written to a MATRIXDATA data type.

The scale component is not considered.

The only supported interpolation type is linear interpolation (LINEAR).

The sceHiPlugAnimePreCalcArg_t structure can be passed to the SCE_HIG_PRE_PROCESS, in a manner similar to sceHiPlugAnime().

The following processes can be called depending on the value specified for *process*.

Table 7-39

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers. Read required data blocks.
SCE_HIG_PRE_PROCESS	Interpolate between matrices. Write result to MATRIXDATA data type.
SCE_HIG_POST_PROCESS	No operation.
SCE_HIG_END_PROCESS	Free internal buffers.

The amount of memory used by the plug-in is shown below for each plug-in block.

Take this into consideration and specify enough memory for the internal buffers using the sceHiMemInit() function.

Table 7-40: Amount of Memory Used

Purpose	Amount Used
Internal buffers	2 (qwords)

The type attributes for this plug-in and the required data are shown below, together with an indication of whether or not the data is writable.

Table 7-41: Plug-in Type Attribute

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_MANIME_PLUG

Table 7-42: Data Type Attribute

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_ANIME_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_KEYFRAME	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_KEYVALUE	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_MATRIXDATA	r/w

Return value

The following detailed error states can be obtained from the sceHiErrStateType variable.

Table 7-43

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- The data type SCE_HIP MATRIXDATA was not found - The data type SCE_HIP ANIME_DATA was not found - The data type SCE_HIP KEYFRAME was not found - The data type SCE_HIP KEYVALUE was not found - The stack area is NULL

sceHiPlugMatrix

Hierarchy matrix plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.7	December 6, 2002

Syntax

```
sceHiErr sceHiPlugMatrix(
  sceHiPlug *plug;           Pointer to hierarchy matrix plug-in block
  int process;);           Plug-in process identifier
```

Calling Conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This plug-in finds a hierarchy matrix by accumulating data from parent IDs that are present in MATRIXDATA and the model local matrix, then writing the result to a local world matrix and light rotation matrix of type BASEMATRIX.

The number of matrices of data type MATRIXDATA must be the same as the number of matrices of data type BASEMATRIX.

The sceHiPlugHrchyPreCalcArg_t structure can be passed to the SCE_HIG_PRE_PROCESS, in a manner similar to sceHiPlugHrchy().

The following processes can be called depending on the value specified for *process*.

Table 7-44

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
	Read required data blocks.
SCE_HIG_PRE_PROCESS	Generate cumulative matrix from MATRIXDATA and write to BASEMATRIX.
SCE_HIG_POST_PROCESS	No operation.
SCE_HIG_END_PROCESS	Free internal buffers.

The amount of memory used by the plug-in is shown below for each plug-in block.

Take this into consideration and specify enough memory for the internal buffers using the sceHiMemInit() function.

Table 7-45: Amount of Memory Used

Purpose	Amount Used
Internal buffers	1 (qword)

The type attributes for this plug-in and the required data are shown below, together with an indication of whether or not the data is writable.

Table 7-46: Plug-in Type Attribute

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_MATRIX_PLUG

Table 7-47: Data Type Attribute

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_MATRIXDATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_BASEMATRIX	r/w

Return Value

The following detailed error states can be obtained from the sceHiErrStateType variable.

Table 7-48

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- The data type SCE_HIP MATRIXDATA was not found - The data type SCE_HIP BASEMATRIX was not found - The stack area is NULL

sceHiPlugMicro

Microcode plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.1	December 6, 2002

Syntax

sceHiErr sceHiPlugMicro(

sceHiPlug **plug*;

Microcode plugin block address

int *process*);

Plugin process ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function DMA transfers microcode to VU1 micro memory and prepares a VU1 double buffer.

Two basic microcodes are switched.

The following processing is called according to the value specified by the process argument.

Table 7-49

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Reserves internal buffers. Generates DMA packets of installed microcode.
SCE_HIG_PRE_PROCESS	Sets registration and attributes of multiple microcodes using <code>sceHiPlugMicroInitArg_t</code> . Switches microcode using <code>sceHiPlugMicroPreCalcArg_t</code> in an actual plugin block argument (<code>HiPlug</code> type member <code>args</code>). If <code>args</code> is <code>NULL</code> , uses the microcode that only handles parallel light source maintained by the library.
SCE_HIG_POST_PROCESS	Registers a Chain using <code>sceHiDMARegist</code> to perform a DMA transfer of DMA packets of microcode. Does not perform the DMA transfer itself. Call <code>sceHiDMASend()</code> ; separately.
SCE_HIG_END_PROCESS	Frees internal buffers and DMA packet buffer.

The amount of memory consumed when this plugin is used is shown below.

Take this amount into account when using the `sceHiDMAInit()`; function in order to specify a size large enough for DMA packet buffers and the packet management area and when using the `sceHiMemInit()`;

function in order to specify a size large enough for internal buffers. This size is consumed for each individual plugin block.

Amount of Memory Consumed

Table 7-50

Use	Amount Consumed
Packet management area	1 qword
DMA packet buffers	4 qword
Internal buffers	2 qword + 2word* num-of-micro code

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes

Table 7-51

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_PLUG

Data Type Attributes

Table 7-52

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_DATA	r/w

Return value

Table 7-53

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
2	SCE_HIG_INVALID_VALUE	Specified Micro code number (ID) is invalid
3	SCE_HIG_INVALID_DATA	- The stack area is NULL - Data types do not match
4	SCE_HIG_FAILURE	- Failed to generate Micro code DMA packet. - Failed to generate Micro data DMA packet. - Data block cannot be found

sceHiPlugMicroGetData

Get MICRO data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugMicroData_t *sceHiPlugMicroGetData(
    sceHiPlug *p);
```

Pointer to plug-in block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a data pointer having a MICRO_DATA data type, which belongs to the MICRO category.

If there is no data, NULL is returned.

Return value

sceHiPlugMicroData_t type MICRO data pointer

sceHiPlugReflect

Reflection plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	December 6, 2002

Syntax

sceHiErr sceHiPlugReflect(

sceHiPlug *plug; Pointer to plug-in block
int process;); Plug-in process identifier

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function produces a reflection mapping effect.

Calculation expressions:

```
reflect=2*(normal,eye)-eye
reflect.x=reflect.x*zoom
reflect.y=reflect.y*zoom
m=sqrt(reflect.x^2+reflect.y^2+(reflect.z+z_shift)^2)
S= reflect.x/(2*m)
T= reflect.y/(2*m)
```

A separate microcode and micro plug-in are required for mapping.

The micro plug-in is inserted in the data format. For specific information about using the reflection plug-in, see the sample source code and data.

Table 7-54

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Generate view matrix.
SCE_HIG_POST_PROCESS	Perform mapping with inserted micro plug-in.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-55

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute**Table 7-56**

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ REFLECT	SCE_HIP_ REFLECT_PLUG
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ MICRO	SCE_HIP_ MICRO_PLUG

Data Type Attribute**Table 7-57**

Repository	Project	Category	Data ID	r/w
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_DATA	r/w

Return value**Table 7-58**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- Plugin type SCE_HIP_MICRO_PLUG cannot be found. - Data type SCE_HIP_MICRO_DATA cannot be found. - The stack area is NULL

sceHiPlugRefract

Refraction plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	December 6, 2002

Syntax

```
sceHiErr sceHiPlugRefract(
sceHiPlug *plug;           Pointer to plug-in block
int process;               Plug-in process identifier
);
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function produces a refraction mapping effect.

Calculation expressions:

```
refract=2*(refract_index*normal.eyeye)+eye
refract.x=refract.x*zoom
refract.y=refract.y*zoom
m=sqrt(refract.x^2+refract.y^2+(refract.z+z_shift)^2)
S= refract.x/(2*m)
T= refract.y/(2*m)
```

A separate microcode and micro plug-in are required for mapping.

The micro plug-in is inserted in the data format. For specific information about using the refraction plug-in, see the sample source code and data.

Table 7-59

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Generate view matrix.
SCE_HIG_POST_PROCESS	Perform mapping with inserted micro plug-in.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-60

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-61

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ REFLECT	SCE_HIP_ REFRACT_PLUG
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ MICRO	SCE_HIP_ MICRO_PLUG

Data Type Attribute

Table 7-62

Repository	Project	Category	Data ID	r/w
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_DATA	r/w

Return value

Table 7-63

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- Plugin type SCE_HIP_MICRO_PLUG cannot be found. - Data type SCE_HIP_MICRO_DATA cannot be found. - The stack area is NULL

Data Type Attribute**Table 7-67**

Repository	Project	Category	Data ID	r/w
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ SHADOW	SCE_HIP_ SHADOWBOX_DATA	r/w
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ SHAPE	SCE_HIP_ BASEMATRIX	r/o

Return value**Table 7-68**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- Data type SCE_HIP_SHADOWBOX_DATA cannot be found. - Data type SCE_HIP_BASEMATRIX cannot be found. - The stack area is NULL

sceHiPlugShadowBoxGetData

Get SHADOWBOX data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugShadowBoxData_t *sceHiPlugShadowBoxGetData(
    sceHiPlug *p);
```

Pointer to plug-in block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a data pointer having a SHADOWBOX_DATA data type, which belongs to the SHADOW category.

Return value

sceHiPlugShadowBoxData_t type SHADOWBOX data pointer

sceHiPlugShadowMap

Shadow map plug-in

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	December 6, 2002

Syntax

```
sceHiErr sceHiPlugShadowMap(
    sceHiPlug *plug;           Pointer to plug-in block
    int process;               Plug-in process identifier
);
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function generates the view screen matrix and texture projection matrix for shadow texture rendering. It produces a shadow effect by using the texture matrix to perform texture mapping from the shadow texture that was rendered.

For INIT_PROCESS, the shadow texture size must be specified with sceHiPlugShadowMapInitArg_t.

For PRE_PROCESS, the ShadowBox plug-in's PRE_PROCESS must be called in advance.

A separate microcode and micro plug-in are required for shadow texture rendering. The micro plug-in is inserted in the data format.

The frame plug-in is used for shadow texture mapping.

Microcode that supports the micro plug-in that is kept by the frame plug-in must be specified.

After the ShadowMap plug-in's PRE_PROCESS, the texture matrix that is in the micro data kept by the ShadowMap plug-in block, must be copied to the micro data that is kept by the frame plug-in block.

This function supports only parallel light source No. 0 within the light information of the micro data that is kept by the ShadowMap.

A suitable drawing environment is required for rendering and mapping.

For specific information about using the shadow map plug-in, see the sample source code and data.

Table 7-69

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Generate view screen matrix for shadow texture rendering. Generate texture matrix for shadow mapping.
SCE_HIG_POST_PROCESS	Perform shadow texture rendering with the inserted micro plug-in. Perform shadow mapping with the frame plug-in.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-70

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-71

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHADOW	SCE_HIP_SHADOWMAP_PLUG

Data Type Attribute

Table 7-72

Repository	Project	Category	Data ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHADOW	SCE_HIP_SHADOWBOX_DATA

Return value

Table 7-73

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
2	SCE_HIG_INVALID_VALUE	plug->args is NULL
3	SCE_HIG_INVALID_DATA	- Plugin type SCE_HIP_MICRO_PLUG cannot be found. - Data type SCE_HIP_MICRO_DATA cannot be found. - The stack area is NULL - Data type SCE_HIP_SHADOWBOX_DATA cannot be found.

Table 7-75: Amount of Memory Consumed

Use	Amount Consumed
Packet management area	Minimum 1 (qword/shape) + 1 (qword)
DMA packet buffers	Maximum 3 (qword/matrix: without skin) Maximum 4 (qword/matrix: with skin) Maximum 3 (qword/shape) Maximum 2 (qword/material: with texture, without material data) Maximum 5 (qword/material: with texture, with material data) Maximum 3 (qword/material: without texture, with material data) Maximum 8 (qword/geometry: maximum 50 vertex is 1 geometry. Amounts over 50 are divided) Maximum 2(qword/skin/4)
Internal buffers	1 word/shape+1word/basematrix + 4 (qword)

Note:

(qword/shape) = qword per shape : Number of qwords per shape

(qword/matrix) = qword per matrix : Number of qwords per matrix

(qword/texture) = qword per texture : Number of qwords per texture

(qword/63vertices/geometry) : Number of qwords per 63 vertices within one geometry

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes**Table 7-76**

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_SHAPE_PLUG

Data Type Attributes**Table 7-77**

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_SHAPE_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_BASEMATRIX	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TEX2D	SCE_HIP_TEX2D_ENV	r/o

Return value**Table 7-78**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	- Failed to allocate internal memory area - Error returned by sceHiDMA*() - VU1 Mem overflow
2	SCE_HIG_INVALID_VALUE	Matrix number (ID) is invalid
3	SCE_HIG_INVALID_DATA	- The stack area is NULL - Data types do not match
4	SCE_HIG_FAILURE	- Error returned by sceHiDMA*() - Data block cannot be found

sceHiPlugShapeGetDataHead

Get Shape data header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugShapeHead_t *sceHiPlugShapeGetDataHead(  
    sceHiPlugShapeHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Shape data item having a SHAPE_DATA data type.
The header pointer obtained from sceHiPlugShapeGetHead() is assigned to **h*.
idx is set to the desired position of the Shape data to be obtained.
The dat structure is used for the return value.

Return value

sceHiPlugShapeHead_t type Header pointer

sceHiPlugShapeGetGeometryColor

Get Geometry vertex color

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugShapeGetGeometryColor(
    sceHiPlugShapeHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Geometry data vertex color having a SHAPE_DATA data type.

The header pointer obtained from sceHiPlugShapeGetGeometryHead() is assigned to **h*.

idx is set to the desired position of the vertex color to be obtained.

Return value

sceVu0FVECTOR type Vertex color pointer

sceHiPlugShapeGetGeometryHead

Get Geometry data header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```

sceHiPlugShapeHead_t *sceHiPlugShapeGetGeometryHead(
    sceHiPlugShapeHead_t *h,                Header pointer
    int idx);                                Index
    
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single header of Geometry data having a SHAPE_DATA data type.

The header pointer obtained from sceHiPlugShapeGetMaterialHead() is assigned to **h*.

idx is set to the desired position of the Geometry data to be obtained.

The geo structure is used for the return value.

Return value

sceHiPlugShapeHead_t type Header pointer

sceHiPlugShapeGetGeometryNormal

Get Geometry normal data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugShapeGetGeometryNormal(
    sceHiPlugShapeHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Geometry normal data item having a SHAPE_DATA data type.

The header pointer obtained from sceHiPlugShapeGetGeometryHead() is assigned to *h.

idx is set to the desired position of the normal data to be obtained.

Return value

sceVu0FVECTOR type	Normal data pointer
--------------------	---------------------

sceHiPlugShapeGetGeometryST

Get Geometry texture coordinates

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```

sceVu0FVECTOR *sceHiPlugShapeGetGeometryST(
  sceHiPlugShapeHead_t *h,           Header pointer
  int idx);                          Index

```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single set of Geometry data texture coordinates having a SHAPE_DATA data type. The header pointer obtained from sceHiPlugShapeGetGeometryHead() is assigned to *h*. *idx* is set to the desired position of the texture coordinates to be obtained.

Return value

sceVu0FVECTOR type Texture coordinate data pointer

sceHiPlugShapeGetGeometryVertex

Get Geometry vertex data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugShapeGetGeometryVertex(
    sceHiPlugShapeHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Geometry vertex data item having a SHAPE_DATA data type.

The header pointer obtained from sceHiPlugShapeGetGeometryHead() is assigned to *h.

idx is set to the desired position of the vertex data item to be obtained.

Return value

sceVu0FVECTOR type Vertex data pointer

sceHiPlugShapeGetHead

Get SHAPE header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugShapeHead_t *sceHiPlugShapeGetHead(  
    sceHiPlug *p;           Pointer to plug-in block  
    sceHiType t);           Data type
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having a SHAPE_DATA or BASEMATRIX data type, which belongs to the SHAPE category.

The top structure is used for the return value.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugShapeHead_t type Header pointer

sceHiPlugShapeGetMaterialAttrib

Get Material attribute

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FMATRIX *sceHiPlugShapeGetMaterialAttrib(
    sceHiPlugShapeHead_t *h);
```

Header pointer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single attribute of Material data having a SHAPE_DATA data type.

The attribute can be ambient, diffuse, or specular.

The header pointer obtained from sceHiPlugShapeGetMaterialHead() is assigned to *h.

If there is no data, NULL is returned.

Return value

sceVu0FMATRIX type Material attribute

sceHiPlugShapeGetMaterialGiftag

Get MaterialGiftag

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiGsGiftag *sceHiPlugShapeGetMaterialGiftag(
    sceHiPlugShapeHead_t *h);
```

Header pointer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Giftag of Material data having a SHAPE_DATA data type.

The header pointer obtained from sceHiPlugShapeGetMaterialHead() is assigned to *h.

If there is no data, NULL is returned.

Return value

sceHiGsGiftag type MaterialGiftag pointer

sceHiPlugShapeGetMaterialHead

Get Material data header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugShapeHead_t *sceHiPlugShapeGetMaterialHead(
    sceHiPlugShapeHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Material data header having SHAPE_DATA data type.

The header pointer obtained from sceHiPlugShapeGetDataHead() is assigned to **h*.

idx is set to the desired position of the Geometry data to be obtained.

The mat structure is used for the return value.

Return value

sceHiPlugShapeHead_t type Header pointer

sceHiPlugShapeGetMatrix

Get matrix data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```

sceHiPlugShapeMatrix_t *sceHiPlugShapeGetMatrix(
    sceHiPlugShapeHead_t *h,                Header pointer
    int idx);                               Index
    
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single matrix data item having a BASEMATRIX data type.

The BASEMATRIX header pointer obtained from sceHiPlugShapeGetHead() is assigned to **h*.

idx is set to the desired position of the matrix data to be obtained.

Return value

sceHiPlugShapeMatrix_t type Matrix data pointer

sceHiPlugShapelInvisible

Set whether shape is visible or invisible

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4	December 6, 2002

Syntax

sceHiErr sceHiPlugShapelInvisible(

sceHiPlug *plug;	Pointer to shape plug-in block
int matidx;	Base matrix ID for displaying shape for which setting is to be made
int flag);	Set to 1 if the shape is to be invisible and 0 if it is to be visible

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets whether a shape to be displayed is visible or invisible.

When the shape is invisible, it is deleted from the list when the DMA transfer is performed (when it is invisible, no DMA transfer is performed).

The setting operation is performed for each individual base matrix. Therefore, when the same shape data is displayed many times using multiple base matrices, the visible / invisible property can be set separately for each base matrix that is displayed.

Return value

Table 7-79

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Error returned by sceHiDMA*()
2	SCE_HIG_INVALID_VALUE	Matrix number (ID) is invalid
3	SCE_HIG_INVALID_DATA	The stack area is NULL
4	SCE_HIG_FAILURE	Error returned by sceHiDMA*()

sceHiPlugShapeMasterChainSetting

Change settings for Shape packet chain

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.2	December 6, 2002

Syntax

```

sceHiErr sceHiPlugShapeMasterChainSetting(
    sceHiPlug *plug;           Pointer to Shape plugin block
    int flag);                 flag value
    
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function changes settings for the Shape packet Master Chain (the packet chain which is attached to the packet chain of displayed objects by CallTag).

The following settings currently can be set.

- Dynamic/Static toggle: Bit 0
 Switches whether the Master Chain is to be created dynamically or statically.
 For Dynamic, set bit 0 of flag to 0. For Static, set bit 0 to 1.
 The default is Dynamic (packet chain is deleted after each transfer).

Return value

Table 7-80

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Error returned by sceHiDMA*()
3	SCE_HIG_INVALID_DATA	The stack area is NULL
4	SCE_HIG_FAILURE	Error returned by sceHiDMA*()

sceHiPlugShare

Share plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.1	December 6, 2002

Syntax

```
sceHiErr sceHiPlugShare(
    sceHiPlug *plug;           Pointer to share plugin block
    int process);              Plugin process ID
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function generates shapes from shared vertices or normal lines according to different coordinate transformations.

It has a pseudo skin deformation effect.

The following processing is called according to the value specified by the *process* argument.

Table 7-81

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Allocates internal buffers. Reads shared vertex data, shared normal line data, shared vertex index data, shared normal line index data, shared vertex link data, shared normal line link data, shape data, and matrix data.
SCE_HIG_PRE_PROCESS	Performs vertex and normal line coordinate transformation from matrix data, generates shapes from link data, and reflects these in shape data.
SCE_HIG_POST_PROCESS	Performs no processing.
SCE_HIG_END_PROCESS	Frees internal buffers.

The amount of memory consumed when this plugin is used is shown below.

Take this amount into account when using the `sceHiMemInit()` function in order to specify a size large enough for the internal buffer.

This size is consumed for each individual plugin block.

Amount of Memory Consumed

Table 7-82

Use	Amount Consumed
Internal buffers	2 (word/geometry)(in share) + 3 (qword)

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes

Table 7-83

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SHARE_PLUG

Data Type Attributes

Table 7-84

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SHARE_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SRCDESTVERTEX	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SRCDESTNORMAL	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_VERTEXINDEX	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_NORMALINDEX	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SHAREVERTEX	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SHARENORMAL	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_SHAPE_DATA	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_BASEMATRIX	r/o

Return value

Table 7-85

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	Failed to allocate internal memory area
3	SCE_HIG_INVALID_DATA	- The stack area is NULL - Vertex data of shape data cannot be obtained - Data types do not match
4	SCE_HIG_FAILURE	Data block cannot be found

sceHiPlugShareGetData

Get SHARE data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugShareData_t *sceHiPlugShareGetData(
    sceHiPlugShareHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single SHARE data item having a SHARE_DATA data type.

The SHARE_DATA header pointer obtained from sceHiPlugShareGetHead() is assigned to *h.

idx is set to the desired position of the SHARE data to be obtained.

Return value

sceHiPlugShareData_t type SHARE data pointer

sceHiPlugShareGetDst

Get SHARE vertex/normal destination data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugShareGetDst(  
    sceHiPlugShareHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single destination data item having a SRCDSTVERTEX or SCR DSTNORMAL data type. The SRCDSTVERTEX or SCR DSTNORMAL header pointer obtained from sceHiPlugShareGetHead() is assigned to *h.

idx is set to the desired position of the destination data item to be obtained.

Return value

sceVu0FVECTOR type SHARE vertex/normal destination data

sceHiPlugShareGetHead

Get SHARE header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugShareHead_t *sceHiPlugShareGetHead(
    sceHiPlug *p;           Pointer to plug-in block
    sceHiType t);           Data type
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having a SHARE_DATA, SRC DST VERTEX, SRC DST NORMAL, VERTEX INDEX, NORMAL INDEX, SHARE VERTEX, or SHARE NORMAL data type, which belongs to the SHARE category.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugShareHead_t type Header pointer

sceHiPlugShareGetIndex

Get SHARE index data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
int *sceHiPlugShareGetIndex(  
    sceHiPlugShareHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single index data item having a VERTEXINDEX or NORMALINDEX data type.
The VERTEXINDEX or NORMALINDEX header pointer obtained from sceHiPlugShareGetHead() is assigned to *h.
idx is set to the desired position of the index data item to be obtained.

Return value

int type SHARE index data pointer

sceHiPlugShareGetShare

Get Share data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugShareData_t *sceHiPlugShareGetShare(
    sceHiPlugShareHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single Share data item having a SHAREVERTEX or SHARENORMAL data type.

The SHAREVERTEX or SHARENORMAL header pointer obtained from sceHiPlugShareGetHead() is assigned to *h.

idx is set to the desired position of the Share data item to be obtained.

Return value

sceHiPlugShareData_t type Share data pointer

sceHiPlugShareGetSrc

Get SHARE vertex/normal source data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceVu0FVECTOR *sceHiPlugShareGetSrc(  
    sceHiPlugShareHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single source data item having a SRC DST VERTEX or SRC DST NORMAL data type. The SRC DST VERTEX or SRC DST NORMAL header pointer obtained from sceHiPlugShareGetHead() is assigned to *h.

idx is set to the desired position of the source data item to be obtained.

Return value

sceVu0FVECTOR type SHARE vertex/normal source data

sceHiPlugSkin

Skin plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	December 6, 2002

Syntax

sceHiErr sceHiPlugSkin(

sceHiPlug **plug*;

Pointer to plugin block

int *process*);

Plugin process identifier

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

By assigning a degree of influence to each vertex as a weighting coefficient due to bones, the Skin plugin can create skin deformation effects using bone hierarchy animations.

Skinning is provided as a library for both VU0 and VU1.

VU0 and VU1 each have the following features and restrictions.

VU0: 1 skin, unlimited bones, 4 weights

VU1: 1 skin, 4 bones, 4 weights

In VU0, the following operations are performed separately according to the bits that are set in the function options of the Skin data (func).

- Low-order 16 bits of func
 - Options for processing the normals of skin
 - 0: Perform normalization when calculating normals
 - 1: Do not perform normalization when calculating normals
 - 2: Reserved
 - 3: Do not calculate normals
- High-order 16 bits of func
 - BASEMATRIX shape options for parent SHAPE to which skin is applied
 - 0: Include only rotation and translation, and perform associated processing for the matrix for which the fourth row is {0,0,0,1}
 - 1: Perform associated processing for a matrix that can be inverted and for which the fourth row is {0,0,0,1}
 - 2: Perform associated processing only for the unit matrix

Table 7-86

Specification Value	Processing Description
SCE_HIG_INIT_PROCESS	Allocates internal buffers. Reads and parses data blocks.
SCE_HIG_PRE_PROCESS	Obtains local world matrix (LW) from local bone matrix (LB) and bone world matrix (BW). When VU0 is used, writes vertices and normal lines of deformation effect from Skin data to Shape data.
SCE_HIG_POST_PROCESS	Does nothing.
SCE_HIG_END_PROCESS	Frees internal buffers.

Memory Consumed

Table 7-87

Use	Amount Consumed
Packet management area	None
DMA packet buffer	None
Internal buffer	2 qwords

Plugin Type Attribute

Table 7-88

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SKIN	SCE_HIP_SKIN_PLUG

Data Type Attribute

Table 7-89

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SKIN	SCE_HIP_SKIN_LB	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SKIN	SCE_HIP_SKIN_LW	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SKIN	SCE_HIP_SKIN_BW	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SKIN	SCE_HIP_SKIN_DATA	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_BASEMATRIX	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_SHAPE_DATA	r/w

Return value

The following detailed error states can be found from the sceHiErrStateType variable.

Table 7-90

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing succeeded
1	SCE_HIG_NO_HEAP	Internal memory allocation failed
3	SCE_HIG_INVALID_DATA	- Data type SCE_HIP_SKIN_LB was not found - Data type SCE_HIP_SKIN_LW was not found - Data type SCE_HIP_SKIN_BW was not found - Data type SCE_HIP_BASEMATRIX was not found - Data type SCE_HIP_SHAPE_DATA was not found - Shape data is invalid - func specification is invalid - The stack area is NULL

sceHiPlugSkinGetBW

Get Skin bone world matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Syntax

```
int *sceHiPlugSkinGetBW(  
    sceHiPlugSkinHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a pointer to a single Skin bone world index having a data type of SKIN_BW.
The SKIN_BW header pointer obtained with the sceHiPlugSkinGetHead() function is assigned to *h.
The position of the index to be obtained is specified for idx.
No distinction is made between a model hierarchy and bone hierarchy.
As a result, the model matrix and bone matrix coexist in BASEMATRIX.
The index obtained here represents the BASEMATRIX index.
The local world matrix that BASEMATRIX holds is the bone world matrix.

Return value

int type Pointer to Skin bone world index

sceHiPlugSkinGetData

Get Skin data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Syntax

```
sceHiPlugSkinData_t *sceHiPlugSkinGetData(
sceHiPlugSkinHead_t *h,           Header pointer
int idx);                         Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single set of Skin data having a data type of SKIN_DATA.

The SKIN_DATA header pointer obtained with the sceHiPlugSkinGetHead() function is assigned to **h*.

The position of the Skin data to be obtained is specified for *idx*.

Return value

sceHiPlugSkinData_t type Pointer to Skin data

sceHiPlugSkinGetHead

Get Skin header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Syntax

```
sceHiPlugSkinHead_t *sceHiPlugSkinGetHead(  
    sceHiPlug *p;           Pointer to plugin block  
    sceHiType t);           Data type
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having a data type of SKIN_LB, SKIN_LW, SKIN_BW, or SKIN_DATA and that belongs to the SKIN category.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugSkinHead_t type Header pointer

sceHiPlugSkinGetLB

Get Skin local bone matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Syntax

```
sceVu0FMATRIX *sceHiPlugSkinGetLB(
    sceHiPlugSkinHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a pointer to a single Skin local bone matrix having a data type of SKIN_LB.

The SKIN_LB header pointer obtained with the sceHiPlugSkinGetHead() function is assigned to **h*.The position of the matrix to be obtained is specified for *idx*.**Return value**

sceVu0FMATRIX type Pointer to Skin local bone matrix

sceHiPlugSkinGetLW

Get Skin local world matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.5.3	July 8, 2002

Syntax

```
sceVu0FMATRIX *sceHiPlugSkinGetLW(  
    sceHiPlugSkinHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a pointer to a single Skin local world matrix having a data type of SKIN_LW.
The SKIN_LW header pointer obtained with the sceHiPlugSkinGetHead() function is assigned to *h.
The position of the matrix to be obtained is specified for idx.

Return value

sceVu0FMATRIX type Pointer to Skin local world matrix

Amount of Memory Consumed**Table 7-92**

Use	Amount Consumed
Packet management area	1 qword
DMA packet buffers	
with clut	8 (qword/texture) + 1 (qword)
without clut	5 (qword/texture) + 1 (qword)
Internal buffers	15 (qword/texture) + 2 (qword)

Note: (qword/texture) = qword per texture : Number of qwords per texture

Specifies the texture buffer that can be used in sceHiPlugTex2dInitArg_t.

The TBP moves from the top down and the CBP moves from the bottom up.

Values can be loaded until they mutually interfere with each other.

If nothing is specified, TBP and CBP are set with 0x1a40 as the top address and 0x4000 as the bottom address.

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes**Table 7-93**

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TEX2D	SCE_HIP_TEX2D_PLUG

Data Type Attributes**Table 7-94**

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TEX2D	SCE_HIP_TEX2D_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TEX2D	SCE_HIP_TEX2D_ENV	r/w

Return value**Table 7-95**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	- Failed to allocate internal memory area - Failed to allocate GS memory area - Error returned by sceHiDMA*()
3	SCE_HIG_INVALID_DATA	- Data types do not match - The stack area is NULL
4	SCE_HIG_FAILURE	- Data block cannot be found - Error returned by sceHiDMA*()

sceHiPlugTex2DGetClut

Get TEX2D CLUT data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
u_int *sceHiPlugTex2DGetClut(
    sceHiPlugTex2DData_t *d);
```

Data pointer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single CLUT data item having a TEX2D_DATA data type.

The single TEX2D_DATA texture data pointer obtained from sceHiPlugTex2DGetData() is assigned to *d.

Return value

u_int type CLUT data pointer

sceHiPlugTex2DGetData

Get TEX2D data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugTex2DData_t *sceHiPlugTex2DGetData(  
    sceHiPlugTex2DHead_t *h,           Header pointer  
    int idx);                          Index
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single texture data item having a TEX2D_DATA data type.
The TEX2D_DATA header pointer obtained from sceHiPlugTex2DGetHead() is assigned to *h.
idx is set to the desired position of the texture data item to be obtained.

Return value

sceHiPlugTex2DData_t type TEX2D data pointer

sceHiPlugTex2DGetEnv

Get TEX2D texture environment data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiGsGiftag *sceHiPlugTex2DGetEnv(
    sceHiPlugTex2DHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single texture environment data item having a TEX2D_ENV data type.

The TEX2D_ENV header pointer obtained from sceHiPlugTex2DGetHead() is assigned to **h*.*idx* is set to the desired position of the texture environment data item to be obtained.**Return value**

sceHiGsGiftag type TEX2D texture environment data pointer

sceHiPlugTex2DGetHead

Get TEX2D header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugTex2DHead_t *sceHiPlugTex2DGetHead(  
    sceHiPlug *p;           Pointer to plug-in block  
    sceHiType t);           Data type
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having a TEX2D_DATA or TEX2D_ENV data type, which belongs to the TEX2D category.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugTex2DHead_t type Header pointer

sceHiPlugTex2DGetTexel

Get TEX2D texel data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
u_int *sceHiPlugTex2DGetTexel(  
    sceHiPlugTex2DData_t *d);
```

Data pointer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single texel data item having a TEX2D_DATA data type.

The single TEX2D_DATA texture data pointer obtained from sceHiPlugTex2DGetData() is assigned to *d.

Return value

u_int type Texel data pointer

sceHiPlugTex2DSize

Get Tex2D plug-in texture size

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	March 29, 2002

Syntax

```
size_t sceHiPlugTex2DSize(
sceHiPlug *plug);
```

Pointer to plug-in block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns the total texture size which the Tex2D plug-in has.

Return value

size_t Texture size

sceHiPlugTim2

Tim2 texture plugin

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.1	December 6, 2002

Syntax

sceHiErr sceHiPlugTim2(

sceHiPlug **plug*; Pointer to Tim2 texture plugin blocks
int *process*); Plugin process identifier

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Performs DMA transfers of 2D image format "TIM2" data to GS local memory.

Download technical information for the TIM2 format from the developer support website.

Specify the texture buffer that can be used in `sceHiPlugTim2InitArg_t`.

The following processes are called that correspond to the value specified in the *process* argument.

Table 7-96

Specified Value	Process
SCE_HIG_INIT_ PROCESS	Reserves internal buffers. Generates DMA packets of decoded texture data for Tim2 format. Sets TBP and CBP of the GS transfer destination using <code>sceHiPlugTim2InitArg_t</code> and specifies resident / non-resident.
SCE_HIG_PRE_ PROCESS	Performs no processing
SCE_HIG_POST_ PROCESS	Performs registration for DMA transferring DMA packets of texture data to GS using <code>sceHiDMARegist()</code> ; . Does not perform the DMA transfer itself. Call <code>sceHiDMASend()</code> ; separately.
SCE_HIG_END_ PROCESS	Frees DMA packet buffer. Frees internal buffers.

Refer to the Tex2D plugin for the amount of memory consumed when using this plugin. The type attributes of this plugin and the type attributes of data which is required are shown.

Plugin type attributes

Table 7-97

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TIM2	SCE_HIP_TIM2PLUG

Data type attributes**Table 7-98**

Repository	Project	Category	Data ID	r/w
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TIM2	SCE_HIP_TIM2_DATA	r/o
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TEX2D	SCE_HIP_TEX2D_ENV	r/w

Return value**Table 7-99**

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
1	SCE_HIG_NO_HEAP	- Cannot allocate area from HiG heap - DMA buffer overflow
2	SCE_HIG_INVALID_VALUE	Specified picture number texture is not in TIM2 data
3	SCE_HIG_INVALID_DATA	- Data is not a TIM2 file - TIM2 file version is different - Type and Data Type written in Data List do not match - Tim2 data address is not specified - Image is not in Tim2 data - Clut format ID in Tim2 data is invalid - The stack area is NULL
4	SCE_HIG_FAILURE	Data specified by Type cannot be found

sceHiPlugTim2GetData

Get TIM2 data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4.3	March 29, 2002

Syntax

```
sceHiPlugTim2Data_t *sceHiPlugTim2GetData(
    sceHiPlugTex2DHead_t *h,           Header pointer
    int idx);                          Index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets a single TIM2 data item having a TIM2_DATA data type.

The TIM2_DATA header pointer obtained from sceHiPlugTim2GetHead() is assigned to **h*.

idx is set to the desired position of the TIM2 data item to be obtained.

Return value

sceHiPlugTim2Data_t type	TIM2 data pointer
--------------------------	-------------------

sceHiPlugTim2GetHead

Get TIM2 header

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	March 29, 2002

Syntax

```
sceHiPlugTim2Head_t *sceHiPlugTim2GetHead(  
    sceHiPlug *p;           Pointer to plug-in block  
    sceHiType t);           Data type
```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the initial header of a data block having a TIM2_DATA data type, which belongs to the TIM2 category.

If the specified data type is not found, NULL is returned.

Return value

sceHiPlugTim2Head_t type Header pointer

sceHiPlugTim2GetName

Get Tim2 plug-in texture filename

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	March 29, 2002

Syntax

```
char *sceHiPlugTim2GetName(
    sceHiPlug *plug;           Pointer to plug-in block
    int idx;);                 Data index
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function gets the filename corresponding to the specified index which the Tim2 plug-in block has.

Return value

char* Pointer to Tim2 texture filename

sceHiPlugTim2GetNPictures

Get number of Tim2 plugin pictures

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4	March 29, 2002

Syntax**sceHiErr sceHiPlugTim2GetNPictures (**

sceHiPlug <i>*plug</i> ;	Pointer to plugin block
int <i>n</i> ;	Data block index
int <i>*num</i>);	Address of int type variable for returning the number of pictures that exist in the corresponding data

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

For the *n*th set of data in the tim2 data list held by *plug*, this function returns the number of pictures contained in that data in *num*.

Note: Only pictures having the same Width/Height/PixelFormat/Clut are supported for Tim2 data with a single data block index (a single Tim2 data file).

Texture chains of Tim2 data that have pictures for which these properties differ are not supported. Clut animation is not supported either.

Return value

Table 7-100

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Data type SCE_HIP_TIM2_DATA cannot be found
4	SCE_HIG_FAILURE	Data type SCE_HIP_TIM2_DATA cannot be found

sceHiPlugTim2Num

Get number of Tim2 plug-in texture files

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	March 29, 2002

Syntax

```
int sceHiPlugTim2Num(
    sceHiPlug *plug);
```

Pointer to plug-in block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function returns the total number of texture files that the Tim2 plug-in block has.

Return value

int Number of Tim2 texture files

sceHiPlugTim2SetData

Set Tim2 plug-in texture data

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.3	March 29, 2002

Syntax

```

sceHiErr sceHiPlugTim2SetData(
    sceHiPlug *plug;           Pointer to plug-in block
    int idx;                   Data index
    u_int *fdata);

```

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Multithread safe (does not depend on interrupt disabled or enabled state)

Description

For the type attribute and the TIM2 data list of the SCE_HIP_TIM2_DATA, this function sets a pointer for TIM2 data which is read separately at the specified index location.

Return value

Table 7-101

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
3	SCE_HIG_INVALID_DATA	Data type SCE_HIP_TIM2_DATA cannot be found
4	SCE_HIG_FAILURE	Data type SCE_HIP_TIM2_DATA cannot be found

sceHiPlugTim2SetPicture

Set Tim2 plugin picture

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libhip	2.4	December 6, 2002

Syntax

```
sceHiErr sceHiPlugTim2SetPicture (
    sceHiPlug *plug;           Pointer to plugin block
    int n;                     Data block index
    int num);                  Picture number
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe (does not depend on interrupt disabled or enabled state)

Description

This function sets the data that *plug* will reference as the *n*th set of texture data so that it is set for picture no. *num*. A texture chain animation can be generated by switching these with appropriate timing.

Note: Only pictures having the same Width/Height/PixelFormat/Clut are supported for Tim2 data with a single data block index (a single Tim2 data file).

Texture chains of Tim2 data that have pictures for which these properties differ are not supported. Clut animation is not supported either.

Return value

Table 7-102

Value	Macro	Result
0	SCE_HIG_NO_ERR	Processing was successful
2	SCE_HIG_INVALID_VALUE	Specified picture number texture is not in TIM2 data.
3	SCE_HIG_INVALID_DATA	- The stack area is NULL - Data is not a TIM2 file - TIM2 file version is different

Chapter 8: VU0 Matrix Arithmetic Library

Table of Contents

Structures	8-3
sceVu0FMATRIX	8-3
sceVu0FVECTOR	8-4
sceVu0IVECTOR	8-5
Functions	8-6
sceVpu0Reset	8-6
sceVu0AddVector	8-7
sceVu0ApplyMatrix	8-8
sceVu0CameraMatrix	8-9
sceVu0ClampVector	8-10
sceVu0ClipAll	8-11
sceVu0ClipScreen	8-12
sceVu0ClipScreen3	8-13
sceVu0CopyMatrix	8-14
sceVu0CopyVector	8-15
sceVu0CopyVectorXYZ	8-16
sceVu0DivVector	8-17
sceVu0DivVectorXYZ	8-18
sceVu0DropShadowMatrix	8-19
sceVu0FTOI0Vector	8-20
sceVu0FTOI4Vector	8-21
sceVu0InnerProduct	8-22
sceVu0InterVector	8-23
sceVu0InterVectorXYZ	8-24
sceVu0InversMatrix	8-25
sceVu0ITOF0Vector	8-26
sceVu0ITOF4Vector	8-27
sceVu0LightColorMatrix	8-28
sceVu0MulMatrix	8-29
sceVu0MulVector	8-30
sceVu0Normalize	8-31
sceVu0NormalLightMatrix	8-32
sceVu0OuterProduct	8-33
sceVu0RotMatrix	8-34
sceVu0RotMatrixX	8-35
sceVu0RotMatrixY	8-36
sceVu0RotMatrixZ	8-37
sceVu0RotTransPers	8-38
sceVu0RotTransPersN	8-39
sceVu0ScaleVector	8-40
sceVu0ScaleVectorXYZ	8-41
sceVu0SubVector	8-42
sceVu0TransMatrix	8-43
sceVu0TransposeMatrix	8-44
sceVu0UnitMatrix	8-45
sceVu0ViewScreenMatrix	8-46

Structures

sceVu0FMATRIX

4x4 matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 26, 2001

Structure

```
typedef float sceVu0FMATRIX[4][4];
```

Description

This is a (float X 4 X 4)-element matrix. The array elements are arranged as follows.

```
| m[0][0] m[1][0] m[2][0] m[3][0] |  
| m[0][1] m[1][1] m[2][1] m[3][1] |  
| m[0][2] m[1][2] m[2][2] m[3][2] |  
| m[0][3] m[1][3] m[2][3] m[3][3] |
```

sceVu0FVECTOR

4-dimensional vector (floating point)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	December 23, 1999

Structure

```
typedef float sceVu0FVECTOR[4];
```

Description

This is a (float X 4)-element vector. The EE is little endian, and the array elements of sceVu0FVECTOR correspond to the vector elements (fields) as follows.

Table 8-1

Array element	Field
a[0]	x
a[1]	y
a[2]	z
a[3]	w

sceVu0VECTOR

4-dimensional vector (integer)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	December 23, 1999

Structure

```
typedef int sceVu0VECTOR[4];
```

Description

This is an (int X 4)-element vector. The EE is little endian, and the array elements of sceVu0VECTOR correspond to the vector elements (fields) as follows.

Table 8-2

Array element	Field
a[0]	x
a[1]	y
a[2]	z
a[3]	w

sceVu0VECTOR can be used with two numeric formats. These formats are fixed point with a 0-bit fractional part and fixed point with a 4-bit fractional part.

Table 8-3

Format	Specifications
Fixed point (32.0)	Integer part: 32 bits, fractional part: 0 bits
Fixed point (28.4)	Integer part: 28 bits, fractional part: 4 bits

Functions

sceVpu0Reset

Reset VU0 and VIF0

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVpu0Reset(void);
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Initializes VU0 and VIF0.

Return value

None

sceVu0AddVector

4-element parallel add (ADD/xyzw)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0AddVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    sceVu0FVECTOR v2)           Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Individually adds the elements of vector *v1* to the corresponding elements of vector *v2* and returns the result in *v0*.

Return value

None

sceVu0ApplyMatrix

Multiply vector by matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0ApplyMatrix(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FMATRIX m0,          Input: Matrix
    sceVu0FVECTOR v1)          Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Multiplies vector *v1* by matrix *m0* from the right and returns the result in vector *v0*. This operation is represented by the following expression.

$$v0 = m0 * v1$$

Return value

None

sceVu0CameraMatrix

Generate world view matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0CameraMatrix(
    sceVu0FMATRIX m,           Output: Matrix (world view coordinates)
    sceVu0FVECTOR p,           Input: Vector (viewpoint)
    sceVu0FVECTOR zd,          Input: Vector (line of sight)
    sceVu0FVECTOR yd)           Input: Vector (normal direction)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains a matrix that transforms the viewpoint *p* to (0,0,0), the line of sight *zd* to (0,0,1), and the normal direction *yd* to (0,1,0) and returns the result in *m*.

Return value

None

sceVu0ClampVector

Clamp vector

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0ClampVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    float min,                  Input: Minimum value
    float max)                  Input: Maximum value
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Clamps each element of vector *v1* using the minimum value *min* and the maximum value *max*, and returns the result in vector *v0*.

Return value

None

sceVu0ClipAll

Check for clipping according to display range

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
int sceVu0ClipAll(
    sceVu0FVECTOR minv,           Input: Minimum value of display range
    sceVu0FVECTOR maxv,           Input: Maximum value of display range
    sceVu0FMATRIX ms,             Input: Matrix (model-screen)
    sceVu0FVECTOR *vm,             Input: Vertex vector pointer
    int n);                       Input: Number of vertices
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Checks whether all of the n vertices specified by vm and n are outside the display range.

Return value

If all of the vertices are outside the display range, 1 is returned.

sceVu0ClipScreen

Check for clipping outside of GS drawing range

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
int sceVu0ClipScreen(
    sceVu0FVECTOR v0)          Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Checks whether or not the vertex vector v0 is inside the GS drawing range.

Return value

When the input vertex is inside the drawing range, 0 is returned.

sceVu0ClipScreen3

Check for clipping outside of GS drawing range

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
int sceVu0ClipScreen3(
    sceVu0FVECTOR v0,          Input: Vector
    sceVu0FVECTOR v1,          Input: Vector
    sceVu0FVECTOR v2)          Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Checks whether all of the vertex vectors *v0*, *v1*, and *v2* are inside the GS drawing range.

Return value

When all input vertices are inside the range, 0 is returned.

sceVu0CopyMatrix

Copy matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0CopyMatrix(
    sceVu0FMATRIX m0,           Output: Matrix
    sceVu0FMATRIX m1);          Input: Matrix
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionCopies matrix *m1* to matrix *m0*.**Return value**

None

sceVu0CopyVector

Copy vector

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0CopyVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1);         Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionCopies vector *v1* to vector *v0*.**Return value**

None

sceVu0CopyVectorXYZ

Copy vector

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0CopyVectorXYZ(
```

```
    sceVu0FVECTOR v0,           Output: Vector
```

```
    sceVu0FVECTOR v1);         Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Copies the elements x, y, and z of vector *v1* to vector *v0*.

The element *w* of vector *v0* is returned directly in vector *v0*.

Return value

None

sceVu0DivVector

Divide

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0DivVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    float q)                     Input: Scalar
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionDivides the vector *v1* by the scalar *q* and returns the result in vector *v0*.**Return value**

None

sceVu0DivVectorXYZ

Divide

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0DivVectorXYZ(
```

```
    sceVu0FVECTOR v0,           Output: Vector
```

```
    sceVu0FVECTOR v1,           Input: Vector
```

```
    float q)                     Input: Scalar
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Divides the elements x , y , and z of vector $v1$ by the scalar q and returns the result in vector $v0$.

The element w of vector $v0$ is returned directly in vector $v0$.

Return value

None

sceVu0DropShadowMatrix

Generate drop shadow projection matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0DropShadowMatrix(
    sceVu0FMATRIX m,           Output: Matrix
    sceVu0FVECTOR lp,          Input: Vector (light source position)
    float a,                    Input: Projection plane of shadow
    float b,                    Input: Projection plane of shadow
    float c,                    Input: Projection plane of shadow
    int mode);                  Input: Light source type
                                0: Parallel light source
                                1: Point light source
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the matrix for projecting the shadow from the light source specified by *lp* and *mode*, onto the plane represented by $ax+by+cz=1$, and returns the result in *m*.

Return value

None

sceVu0FTOI0Vector

Floating point -> fixed point with 0-bit fractional part

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0FTOI0Vector(
    sceVu0IVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1);         Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Converts each element of floating-point vector *v1* to a fixed-point number with 0-bit fractional part and returns the result in *v0*.

Return value

None

sceVu0FTOI4Vector

Floating point -> fixed point with 4-bit fractional part

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0FTOI4Vector(
    sceVu0IVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1);         Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Converts each element of floating-point vector *v1* to a fixed-point number with 4-bit fractional part and returns the result in *v0*.

Return value

None

sceVu0InnerProduct

Inner product of vectors

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
float sceVu0InnerProduct(
    sceVu0FVECTOR v0,           Input: Vector
    sceVu0FVECTOR v1)          Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the inner product of vectors *v0* and *v1*.

Return value

Returns the inner product.

sceVu0InterVector

Generate interpolation vector

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0InterVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    sceVu0FVECTOR v2,           Input: Vector
    float t)                     Input: Interpolation parameter
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the interpolation vector from vectors *v1* and *v2* and parameter *t* and returns the result in *v0*.

This operation is represented by the following expression.

$$v0 = v1 * t + v2 * (1 - t)$$

Return value

None

sceVu0InterVectorXYZ

Generate interpolation vector

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax**void sceVu0InterVectorXYZ(**

sceVu0FVECTOR <i>v0</i> ,	Output: Vector
sceVu0FVECTOR <i>v1</i> ,	Input: Vector
sceVu0FVECTOR <i>v2</i> ,	Input: Vector
float <i>t</i>)	Input: Interpolation parameter

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionObtains the interpolation vector from vectors *v1* and *v2* and parameter *t* and returns the result in *v0*.

This operation is represented by the following expression.

$$v0 = v1 * t + v2 * (1 - t)$$

However, the element *w* of vector *v0* is returned directly in vector *v0*.**Return value**

None

sceVu0InversMatrix

Generate inverse matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0InversMatrix(
    sceVu0FMATRIX m0,           Output: Matrix
    sceVu0FMATRIX m1);         Input: Matrix (regular matrix)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionObtains the inverse of matrix *m1* and returns the result in matrix *m0*.*m1* must be a regular matrix (rotation or translation matrix).**Return value**

None

sceVu0ITOF0Vector

Fixed point with 0-bit fractional part -> floating point

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0ITOF0Vector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0IVECTOR v1);         Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Converts each element of fixed-point vector *v1* (with 0-bit fractional part) to a floating-point number and returns the result in *v0*.

Return value

None

sceVu0ITOF4Vector

Fixed point with 4-bit fractional part -> floating point

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0ITOF4Vector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0IVECTOR v1);         Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Converts each element of fixed-point vector *v1* (with 4-bit fractional part) to a floating-point number and returns the result in *v0*.

Return value

None

sceVu0LightColorMatrix

Generate light color matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

void sceVu0LightColorMatrix(

sceVu0FMATRIX <i>m</i> ,	Output: Matrix
sceVu0FVECTOR <i>c0</i> ,	Input: Vector (light source color 0)
sceVu0FVECTOR <i>c1</i> ,	Input: Vector (light source color 1)
sceVu0FVECTOR <i>c2</i> ,	Input: Vector (light source color 2)
sceVu0FVECTOR <i>a</i>);	Input: Vector (ambient light color)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the light color matrix from light source colors *c0*, *c1*, and *c2* and ambient light color *a* and returns the result in *m*.

Return value

None

sceVu0MulMatrix

Multiply matrices

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0MulMatrix(
    sceVu0FMATRIX m0,           Output: Matrix
    sceVu0FMATRIX m1,           Input: Matrix
    sceVu0FMATRIX m2)           Input: Matrix
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Multiplies matrix *m2* by matrix *m1* from the right and returns the result in *m0*. This operation is represented by the following expression.

$$m0 = m1 * m2$$

Return value

None

sceVu0MulVector

4-element multiply (MUL/xyzw)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0MulVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    sceVu0FVECTOR v2)           Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Individually multiplies the elements of vector *v1* by the corresponding elements of vector *v2* and returns the result in *v0*.

Return value

None

sceVu0Normalize

Normalize vector

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0Normalize(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1)          Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Normalizes vector *v1* and returns the result in *v0*.

Return value

None

sceVu0NormalLightMatrix

Generate normal light matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

void sceVu0NormalLightMatrix(

sceVu0FMATRIX <i>m</i> ,	Output: Matrix
sceVu0FVECTOR <i>l0</i> ,	Input: Vector (light source 0 direction)
sceVu0FVECTOR <i>l1</i> ,	Input: Vector (light source 1 direction)
sceVu0FVECTOR <i>l2</i>)	Input: Vector (light source 2 direction)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the normal light matrix from light sources *l0*, *l1*, and *l2* and returns the result in *m*.

Return value

None

sceVu0OuterProduct

Outer product of vectors

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0OuterProduct(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    sceVu0FVECTOR v2)           Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionObtains the outer product of vectors *v1* and *v2* and returns the result in *v0*.**Return value**

None

sceVu0RotMatrix

Rotate matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0RotMatrix(
    sceVu0FMATRIX m0,           Output: Matrix
    sceVu0FMATRIX m1,           Input: Matrix
    sceVu0FVECTOR rot);         Input: x-, y-, and z-axis rotation angles (valid range: -pi to +pi)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the rotation matrix for rotation about the Z-axis from *rot[2]*, the rotation matrix for rotation about the Y-axis from *rot[1]*, and the rotation matrix for rotation about the X-axis from *rot[0]*, then sequentially multiplies matrix *m1* by these matrices from the left, and returns the result in matrix *m0*.

Return value

None

sceVu0RotMatrixX

Rotate about X-axis

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0RotMatrixX(
    sceVu0FMATRIX m0,           Output: Matrix
    sceVu0FMATRIX m1,           Input: Matrix
    float rx);                  Input: Rotation angle (valid range: -pi to +pi)
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the rotation matrix for rotation about the X-axis from the specified rotation angle *rx*, then multiplies matrix *m1* by this matrix from the left, and returns the result in matrix *m0*.

Return value

None

sceVu0RotMatrixY

Rotate about Y-axis

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax**void sceVu0RotMatrixY(****sceVu0FMATRIX *m0*,** Output: Matrix**sceVu0FMATRIX *m1*,** Input: Matrix**float *ry*);** Input: Rotation angle (valid range: -pi to +pi)**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the rotation matrix for rotation about the Y-axis from the specified rotation angle *ry*, then multiplies matrix *m1* by this matrix from the left, and returns the result in matrix *m0*.

Return value

None

sceVu0RotMatrixZ

Rotate about Z-axis

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax**void sceVu0RotMatrixZ(****sceVu0FMATRIX *m0*,** Output: Matrix**sceVu0FMATRIX *m1*,** Input: Matrix**float *rz*);** Input: Rotation angle (valid range: -pi to +pi)**Calling conditions**

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the rotation matrix for rotation about the Z-axis from the specified rotation angle *rz*, then multiplies matrix *m1* by this matrix from the left, and returns the result in matrix *m0*.

Return value

None

sceVu0RotTransPers

Perspective transformation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0RotTransPers(
    sceVu0IVECTOR v0,           Output: Vector representing screen coordinates
    sceVu0FMATRIX m0,          Input: Perspective transformation matrix
    sceVu0FVECTOR v1,          Input: Vector representing the vertex
    int mode);                  Input: Format specification of output coordinate values
                                v0[2] and v0[3]
                                0: Fixed point with 4-bit fractional part
                                1: Fixed point with 0-bit fractional part
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Performs perspective transformation to screen coordinates on the vertex specified by vector *v1*, using matrix *m0*, and returns the result in vector *v0*.

The numeric format of the *v0[0]* and *v0[1]* output values is 32-bit signed fixed-point with a 4-bit fractional part. The numeric format of the *v0[2]* and *v0[3]* output values varies according to the *mode* specification. When *mode*=0, the format is 32-bit unsigned fixed-point with 4-bit fractional part. When *mode*=1, the format is 32-bit unsigned fixed-point with 0-bit fractional part. When XYZF2 and XYZF3 are used with GIF PACKED mode, specifying *mode*=0 is useful because the integer part is cut out during packing.

Return value

None

sceVu0RotTransPersN

Perspective transformation

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

void sceVu0RotTransPersN(
 sceVu0IVECTOR **v0*, Output: Pointer to vector representing screen coordinates
 sceVu0FMATRIX *m0*, Input: Matrix
 sceVu0FVECTOR **v1*, Input: Pointer to vector representing vertex
 int *n*, Input: Number of vertices
 int *mode*); Input: Format specification of output coordinate values
 v0[2] and *v0[3]*
 0: Fixed point with 4-bit fractional part
 1: Fixed point with 0-bit fractional part

Calling conditions

- Can be called from an interrupt handler
- Can be called from a thread
- Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Performs perspective transformation to screen coordinates on the *n* vertices specified by *v1*, using matrix *m0*, and returns the result in the area pointed to by *v0*.

The numeric format of the **v0[0]* and **v0[1]* output values is 32-bit signed fixed-point with a 4-bit fractional part. The numeric format of the **v0[2]* and **v0[3]* output values varies according to the mode specification. When *mode*=0, the format is 32-bit unsigned fixed-point with 4-bit fractional part. When *mode*=1, the format is 32-bit unsigned fixed-point with 0-bit fractional part. When XYZF2 and XYZF3 are used with GIF PACKED mode, specifying *mode*=0 is useful because the integer part is cut out during packing.

Return value

None

sceVu0ScaleVector

Multiply vector by a scalar (MULx/xyzw)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0ScaleVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    float t)                     Input: Scalar
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Multiplies the vector *v1* by the scalar *t* and returns the result in *v0*.

Return value

None

sceVu0ScaleVectorXYZ

Multiply vector by a scalar (MULx/xyz)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

void sceVu0ScaleVectorXYZ(

sceVu0FVECTOR <i>v0</i> ,	Output: Vector
sceVu0FVECTOR <i>v1</i> ,	Input: Vector
float <i>t</i>)	Input: Scalar

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Multiplies the elements *x*, *y*, and *z* of vector *v1* by the scalar *t* and returns the result in vector *v0*.

The element *w* of vector *v1* is returned directly in vector *v0*.

Return value

None

sceVu0SubVector

4-element parallel subtract (SUB/xyzw)

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0SubVector(
    sceVu0FVECTOR v0,           Output: Vector
    sceVu0FVECTOR v1,           Input: Vector
    sceVu0FVECTOR v2)           Input: Vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Individually subtracts the elements of vector *v2* from the corresponding elements of vector *v1* and returns the result in *v0*.

Return value

None

sceVu0TransMatrix

Translate matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0TransMatrix(
    sceVu0FMATRIX m0,           Output: Matrix
    sceVu0FMATRIX m1,           Input: Matrix
    sceVu0FVECTOR tv);          Input: Translation vector
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionTranslates matrix *m1* using vector *tv* and returns the result in *m0*.**Return value**

None

sceVu0TransposeMatrix

Generate transposed matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0TransposeMatrix(  
    sceVu0FMATRIX m0,           Output: Matrix  
    sceVu0FMATRIX m1)           Input: Matrix
```

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the transposed matrix of matrix *m1* and returns the result in *m0*.

Return value

None

sceVu0UnitMatrix

Generate unit matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

```
void sceVu0UnitMatrix(
    sceVu0FMATRIX m0);
```

Output: Matrix

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

DescriptionReturns a 4x4 unit matrix in matrix *m0*.**Return value**

None

sceVu0ViewScreenMatrix

Generate view screen matrix

<i>Library</i>	<i>Introduced</i>	<i>Documentation last modified</i>
libvu0	1.1	March 29, 2002

Syntax

void sceVu0ViewScreenMatrix(

sceVu0FMATRIX <i>m</i> ,	Output: Matrix
float <i>scrz</i> ,	Input: (distance to screen)
float <i>ax</i> ,	Input: (X-direction aspect ratio)
float <i>ay</i> ,	Input: (Y-direction aspect ratio)
float <i>cx</i> ,	Input: (X-coordinate of center of screen)
float <i>cy</i> ,	Input: (Y-coordinate of center of screen)
float <i>zmin</i> ,	Input: (Z-buffer minimum value)
float <i>zmax</i> ,	Input: (Z-buffer maximum value)
float <i>nearz</i> ,	Input: (Z of near clipping plane)
float <i>farz</i>)	Input: (Z of far clipping plane)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe (does not depend on interrupt disabled or enabled state)

Description

Obtains the view screen matrix using the specified parameters and returns the result in *m*.

Return value

None