PlayStation®2 EE Library Reference
Release 2.4.2

Graphics Libraries

Publication date: December 2001

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

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

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

Summary Table of Contents

About This Manual

This is the Runtime Library Release 2.4.2 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

Chapter 7: High Level Graphics Plugin Library

- A description of the sceHiPlugShapeMasterChainSetting() function has been added.

Related Documentation

Library specifications for the IOP can be found in the *PlayStation®2 IOP Library Reference* manuals and the *PlayStation®2 IOP Library Overview* manuals.

Note: the Developer Support Web site posts current developments regarding the Libraries and also provides notice of future documentation releases and upgrades.

Typographic Conventions

Certain Typographic Conventions are used throughout this manual to clarify the meaning of the text:

Convention	Meaning
`courier`	Indicates literal program code.
italic	Indicates names of arguments and structure members (in structure/function definitions only).
medium bold	Indicates data types and structure/function names (in structure/function definitions only).
blue	Indicates a hyperlink.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In North America:	*In North America:*
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd. Foster City, CA 94404, U.S.A. Tel: (650) 655-8000	E-mail: PS2_Support@playstation.sony.com Web: http://www.devnet.scea.com/ Developer Support Hotline: (650) 655-5566 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In Europe:	*In Europe:*
Attn: Production Coordinator Sony Computer Entertainment Europe 30 Golden Square London W1F 9LD, U.K. Tel: +44 (0) 20 7859-5000	E-mail: ps2_support@scee.net Web: https://www.ps2-pro.com/ Developer Support Hotline: +44 (0) 20 7859-5777 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT)

Chapter 1: Basic Graphics Library
Table of Contents

Chain Management Structure

sceGpChain

Chain management structure

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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.

Register Unions

sceGpPack

P-format register union

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpChkChainOtSize(

*r***)** Resolution

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

int sceGpChkNumPtoV(

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

Description

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

Return value

Number of vertices

sceGpChkPacketSize

Check packet size

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

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

Chain Manipulation Functions

sceGpAddChain

Register child chain

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

sceGpCallChain

Register child chain call

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

sceGpInitChain

Initialize chain

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by ordering table

sceGpKickChain

Transfer chain

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

0: When transfer started

-1: When processing for starting the transfer failed because the DMA channel to be used for the transfer was busy

sceGpPrintChain

Output chain contents to console

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

sceGpResetChain

Reset chain ordering table

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

sceGpSetStartLevel

Set intermediate start within chain

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

void sceGpSetStartLevel(

sceGpChain ******chain***,**	Address of chain management structure
int *level***)**	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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

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.)

Return value

None

Packet Control Functions

sceGpAddPacket

Register packet in chain

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

sceGpCopyPacket

Copy packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Copied memory size (in quad words)

sceGpInsertPacket

Add packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

void sceGpSetDefaultCtxt(

int *ctxt***)** Default context (0: CTXT1, 1: CTXT2)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Packet Initialization Functions

sceGpInitAd

Initialize general-purpose register setting packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by packet

sceGpInitAlphaEnv

Initialize alpha environment setting packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by the packet

sceGpInitCall

Initialize call packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by the packet

sceGpInitLoadImage

Initialize image transfer packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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, trxposaddr, trxregaddr, trxdiraddr, giftag2, part of dmaref, part of dmanext, giftag3, texflush, and texflushaddr.

Return value

Amount of memory (in quad words) to be used by the packet

sceGpInitLoadTexelClut

Initialize texture transfer packet with CLUT

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by the packet

sceGpInitPacket

Initialize packet (general)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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 1-1

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

Amount of memory (in quad words) to be used by the packet

sceGpInitPrimP

Initialize P-format drawing packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by the packet

sceGpInitPrimR

Initialize R-format drawing packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by the packet

sceGpInitRef

Initialize ref packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by the packet

sceGpInitTexEnv

Initialize texture environment setting packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Amount of memory (in quad words) to be used by the packet

sceGpInitTexEnvMipmap

Initialize texture environment setting packet with MipMap

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

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

Table 1-2

Setting Value	Output Color
SCE_GP_ALPHA_NOP	Source color asis
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)

Setting Value	Output Color
SCE_GP_ALPHA_ADD_CS_AD	Color obtained by multiplying source color by Ad and adding it (to destination 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)>>7.

The functions that can by 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 packet contents after the setting value is set.

Zero will be set for the values of the PABE and FBA registers. 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, look out for contention if context switching is used.

Return value

None

sceGpSetAlphaEnvFunc

Set alpha blending function of alpha environment setting packet

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

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

Table 1-3

Setting Value	Output Color
SCE_GP_ALPHA_NOP	Source color asis
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)>>7.

The functions that can by 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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

void sceGpSetLoadTexelClut(

sceGpLoadTexelClut ******p***,**	Address of packet to be set
sceGpTextureArg ******texarg***,**	Texture parameter structure
void ******tsrcaddr***,**	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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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.

The texture color component TCC value 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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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.

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

Return value

None

Index Acquisition Functions (for drawing packets)

sceGpIndexQ#_R

Get Q index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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

Index for which *n*-th Q value should be set

sceGpIndexQ_R

Get Q index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

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

Return value

Index for which *n*-th Q value should be set

sceGpIndexRgba

Get RGBA index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

unsigned int sceGpIndexRgba(

unsigned int *type***,**	Packet type
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

Description

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

Return value

Index for which *n*-th RGBA value should be set

sceGpIndexRgba#

Get RGBA index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

This 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

Index for which *n*-th RGBA value should be set

sceGpIndexSt#_R

Get ST index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

This 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 value

Index for which *n*-th ST value should be set

sceGpIndexSt_R

Get ST index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

unsigned int sceGpIndexSt_R(

unsigned int *type***,**	Packet type
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

Description

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

Return value

Index for which *n*-th ST value should be set

sceGpIndexStq#_P

Get STQ index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

This 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 value

Index for which *n*-th STQ value should be set

sceGpIndexStq_P

Get STQ index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

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

Return value

Index for which *n*-th STQ value should be set

sceGpIndexUv

Get UV index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

unsigned int sceGpIndexUv(

unsigned int *type***,**	Packet type
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

Description

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

Return value

Index for which *n*-th UV value should be set

sceGpIndexUv#

Get UV index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

This 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 value

Index for which *n*-th UV value should be set

sceGpIndexXyzf

Get XYZF index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

unsigned int sceGpIndexXyzf(

unsigned int *type***,**	Packet type
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

Description

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

Return value

Index for which *n*-th XYZF value should be set

sceGpIndexXyzf#

Get XYZF index

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

Description

This 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

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetAbe(

p,	Pointer to packet for which value is to be set
v)	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

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 CTXT value (macro function)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

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

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetFog(

p,	Pointer to packet for which value is to be set
v)	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

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetRgb(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetRgba(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

void sceGpSetRgbaFM(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetStq_P(

p,	Pointer to packet for which value is to be set
k,	Index
s,	S value
t,	T value
q)	Q value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetStq_R(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetUv(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetXy(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetXyz(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

sceGpSetXyzf

Set XYZF value (macro function)

Library	*Introduced*	*Documentation last modified*
libgp	2.4.1	November 5, 2001

Syntax

#define sceGpSetXyzf(

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

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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)

Chapter 2: GS Basic Library
Table of Contents

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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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 *bgcolor;*	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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

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

Syntax

int sceGsExecLoadImage(

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

Calling conditions

Can be called from a thread

Not multithread safe

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

Exit conditions

0**:** Normal termination

-1: Timeout

sceGsExecStoreImage

Execute StoreImage

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

Syntax

int sceGsExecStoreImage(

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

Calling conditions

Can be called from a thread

Not multithread safe

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

Exit conditions

0**:** Normal termination

-1: Timeout

sceGsGetGParam

Get library system information

Library	*Introduced*	*Documentation last modified*
libgraph	1.6	March 26, 2001

Syntax

sceGsGParam *sceGsGetGParam(void)

Calling conditions

Can be called from a thread

Not multithread safe

Description

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

Return value

Pointer to structure currently being used by the system.

sceGsGetIMR/isceGsGetIMR

Get interrupt mask

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

Syntax

u_long sceGsGetIMR(void)

u_long isceGsGetIMR(void)

Calling conditions

Can be called from a thread

Not multithread safe

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

IMR register value previously set by sceGsPutIMR/isceGsPutIMR

sceGsPutDispEnv

Initialize the display environment

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

Syntax

void sceGsPutDispEnv(

sceGsDispEnv **disp)* Address of structure used to set up the display environment

Calling conditions

Can be called from a thread

Not multithread safe

Description

Transfers the contents of the structure pointed to by *disp* to the corresponding GS registers.

Return value

None

sceGsPutDrawEnv

Initialize the drawing environment

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

Syntax

int sceGsPutDrawEnv(

sceGifTag ****giftag*)** Starting address of drawing environment settings data

Calling conditions

Can be called from a thread

Not multithread safe

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

Termination state

0**:** Normal termination

-1**:** Timeout (when preceding Ch-2 DMA has not finished)

sceGsPutIMR/isceGsPutIMR

Set up interrupt mask

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

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

Can be called from a thread

Not multithread safe

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

IMR register value previously set by sceGsPutIMR/isceGsPutIMR

sceGsResetGraph

Initialize the GS

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

Syntax

void sceGsResetGraph(

short *mode,*	Reset mode 0: General reset 1: Drawing reset
short *inter,*	Interlace/non-interlace settings (valid only when mode==0) 0**:** Non-interlace 1**:** Interlace
short *omode,*	Video signal format setting (valid only when mode==0) 2**:** NTSC 3**:** PAL
short *ffmode***)**	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

Can be called from a thread

Not multithread safe

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.

Return value

None

sceGsResetPath

Initialize datapath device

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

Syntax

void sceGsResetPath(void)

Calling conditions

Can be called from a thread

Not multithread safe

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

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

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 a thread

Not multithread safe

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

Size of the sceGsAlphaEnv structure

sceGsSetDefAlphaEnv2

Generate alpha blending environment data

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

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 a thread

Not multithread safe

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

Size of the sceGsAlphaEnv2 structure

sceGsSetDefClear

Generate buffer clear data

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

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 a thread

Not multithread safe

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

Size of the sceGsClear structure (in words)

sceGsSetDefClear2

Generate buffer clear data

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

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 a thread

Not multithread safe

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

Size of the sceGsClear structure (in words)

sceGsSetDefDBuff

Set up double buffering

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

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 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

Can be called from a thread

Not multithread safe

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/FRAME 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

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

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

Can be called from a thread

Not multithread safe

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/FRAME 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

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

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

Can be called from a thread

Not multithread safe

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;*

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 = (50+*dy*) <<12

display2.DX = 0x27c + (*dx**(2560/*w*))

bgcolor = 0

PAL mode:

```
pmode = 0x66
smode2 = 3
dispfb2.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

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

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 a thread

Not multithread safe

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

Size of the sceGsDrawEnv1 structure (in words)

sceGsSetDefDrawEnv2

Generate drawing environment data

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

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 a thread

Not multithread safe

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

Size of the sceGsDrawEnv2 structure (in words)

sceGsSetDefLoadImage

Set up LoadImage information

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

Syntax

int sceGsSetDefLoadImage(

sceGsLoadImage ******lp,*	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 a thread

Not multithread safe

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

Size of the sceGsLoadImage structure (in words)

sceGsSetDefStoreImage

Set up StoreImage information

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

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 a thread

Not multithread safe

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* x *h* x 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

Size of the sceGsStoreImage structure

sceGsSetDefTexEnv

Generate texture environment settings

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

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*	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 GS 3**:** Load from *cbp* and enter *cbp* value in the CBP1 register of the GS 4**:** 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 a thread

Not multithread safe

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

Size of the sceGsTexEnv structure (in words)

sceGsSetDefTexEnv2

Generate texture environment settings

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

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 GS 3**:** Load from *cbp* and enter *cbp* value in the CBP1 register of the GS 4**:** 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 a thread

Not multithread safe

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

Size of the sceGsTexEnv2 structure (in words)

sceGsSetHalfOffset

Generate drawing offset data

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

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 a thread

Not multithread safe

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

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

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 a thread

Not multithread safe

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

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

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 a thread

Not multithread safe

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

Termination status

0: Normal termination

-1: Timeout (when a preceding Ch.2 DMA has not finished)

sceGsSwapDBuffDc

Swap double buffers

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

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 a thread

Not multithread safe

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

Termination status

0: Normal termination

-1: Timeout (when a preceding Ch.2 DMA has not finished)

sceGsSyncPath

Wait for data transfer to finish

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

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 a thread

Not multithread safe

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, before completion, 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

Exit status.

When *mode* == 0,

0**:** Normal termination
Negative value**:** Abnormal termination (timeout)

When *mode* == 1,

0**:** Wait condition is released
Positive value**:** Busy status of the following devices

31	4	3	2	1	0
	GIF	VU1	VIF1	DMA2	DMA1

0**:** idle

1**:** busy

sceGsSyncV

Wait for sync with V-Blank

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

Syntax

int sceGsSyncV(

int *mode***)** Reserved (specify 0)

Calling conditions

Can be called from a thread

Not multithread safe

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

FIELD information for interlaced mode

0: Even field

1: Odd field

A 1 is always returned for non-interlaced mode

sceGsSyncVCallback

Set up VSync callback

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

Syntax

int *sceGsSyncVCallback(

int *(*func)***(int))** Entry address for callback function

Calling conditions

Can be called from a thread

Not multithread safe

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

The entry address for the previously set up callback function.

Chapter 3: DMA Packet Management Services
Table of Contents

Functions

sceHiDMADel_Chain

Delete chain

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

This function deletes a DMA Chain id that was previously created.

The deleted DMA Buffer area becomes an unused Dead area.

To reuse the area, call sceHiDMAPurge().

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The relevant chain does not exist

sceHiDMAGet_BufferPtr

Get packet buffer pointer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

This function returns the current buffer pointer value in *addr*.

Return value

SCE_HIG_NO_ERR Processing was successful

sceHiDMAGet_ChainAddr

Get chain address

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

This function returns the starting address of the chain indicated by *id*.

Note that this will not be a valid value after sceHiDMAPurge() is executed.

Use this function to obtain the address each time at necessary locations.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	

sceHiDMAInit

Initialize DMA

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Memory allocation failed

sceHiDMAInit_DBuf

Initialize double buffer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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);
```

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_CallID

Create packet for calling chain with ID

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

This function creates an instruction packet for calling a previously created chain.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The relevant chain does not exist
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_CallPtr

Create packet for calling a chain with a pointer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

This function creates an instruction packet for calling a previously created chain located at *ptr*.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_ChainEnd

End chain creation

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Memory allocation failed
SCE_HIG_FAILURE	Chain not started
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_ChainStart

Start chain creation

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_ChainStart(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	A chain that has not been ended already exists

sceHiDMAMake_ContinueMicro

Restart execution of micro code

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_ContinueMicro(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function creates a packet for restarting a micro code program that has been stopped.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_DBufEnd

End creation of double buffer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_DBufEnd(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function declares the end of transfer instruction packet creation in the Double Buffer.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	MakeDBufStart has not been executed

sceHiDMAMake_DBufStart

Start Double Buffer creation

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_DBufStart(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	No Double Buffer settings were found

sceHiDMAMake_DynamicChainEnd

End dynamic chain creation

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiDMAMake_DynamicChainEnd(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing succeeded
SCE_HIG_FAILURE	Processing failed

sceHiDMAMake_DynamicChainStart

Start dynamic chain creation

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiDMAMake_DynamicChainStart(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing succeeded
SCE_HIG_FAILURE	Processing failed

sceHiDMAMake_ExecMicro

Start execution of micro code

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_ExecMicro(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LoadGS

Transfer data to the GS

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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*.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LoadGSLump

Transfer data to GS

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

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

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.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LoadImm

Transfer data with an immediate value

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

This function creates an instruction packet for transferring the qword immediate value *imm* to the VU1 address specified by *addr*.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LoadMicro

Transfer micro code

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LoadPtr

Transfer data with a pointer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_LoadPtr(

u_int *****_addr_,	VU1 transfer-destination address
u_int _ptr_,	EE transfer-source address
size_t _qsize_**)**	Transfer volume (qword size)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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*.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LoadStep

Transfer data with offset

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

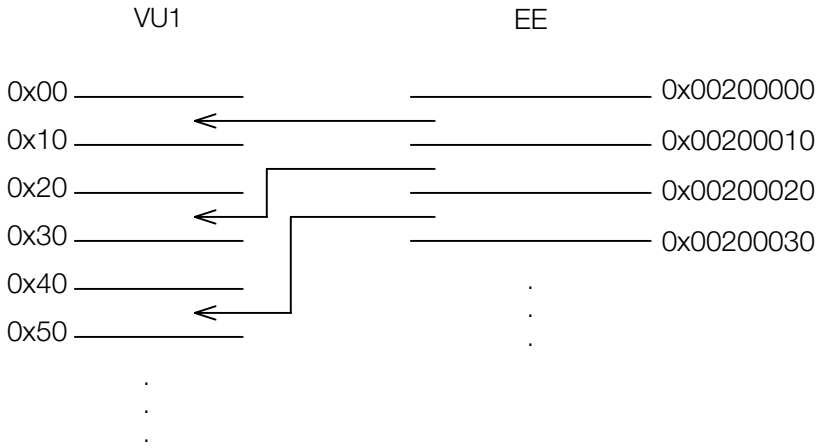
Description

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)
```

Figure 3-1



Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_Lump

Create packet with immediate data string

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

See sceHiDMAMake_LumpStart().

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	Packet creation has not been started
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LumpEnd

End packet creation with immediate data string

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_LumpEnd(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

See the description of **sceHiDMAMake_LumpStart()**.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	Attempt was made to end packet creation that had not been started
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_LumpStart

Start packet creation with immediate data string

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	Attempt to start packet creation when packet creation has not been ended
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAMake_WaitMicro

Wait for micro code

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMAMake_WaitMicro(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function creates an instruction packet that waits for the micro code program to stop.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Buffer overflow

sceHiDMAPurge

Reconfigure chain buffer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

Syntax

sceHiErr sceHiDMAPurge(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function exists only for compatibility with earlier versions. Currently, there is no need to call it.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiDMARegist

Register chain

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMARegist(

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

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.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	Relevant chain does not exist
SCE_HIG_FAILURE	Relevant chain was already registered
SCE_HIG_INVALID_VALUE	Chain has been destroyed

sceHiDMASend

DMA transfer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiDMASend (void);

Calling conditions

Cannot be called from an interrupt handler

Can be called from a thread

Not multithread safe

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.

```
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();
     sceGsSyncV(0);
    }
}
```

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiDMASendI

Perform DMA transfer

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiDMASendI(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function enables sceHiDMASend() to be called from within an interrupt handler.

Return value

SCE_HIG_NO_ERR	Processing succeeded

sceHiDMASet_BufferPtr

Set buffer pointer

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiDMASwap

Swap DMA registration buffer

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiDMASwap(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function swaps the buffer used for chains for which transfer was reserved using the sceHiDMARegist() function. It enables DMA chains to be created and registered in the background of DMA transfer processing.

Return value

SCE_HIG_NO_ERR	Processing succeeded

sceHiDMAWait

Wait for end of DMA transfer

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiDMAWait(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing succeeded

Chapter 4: High Level Graphics Library
Table of Contents

Structures

sceHiData

Data block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

Structure

typedef struct _sceHiData {

sceHiType *type;*	Data type of the data
char *count;*	Number of plugins that are currently using this data (part of system management information)
char *reserve[3];*	Reserved area for future expansion
u_int *size;*	Data size (byte size)
u_int *data[1];*	Data (variable-length array)

} sceHiData;

Description

This is the data block type.

sceHiErr

Error

Library	*Introduced*	*Documentation last modified*
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 plugin 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 plugins.

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 plugin of plugin where ERROR occurred
    sceHiPlug *plug;         Plugin where ERROR occurred
    int process;             Process number where ERROR occurred
    sceHiType type;          Type attribute of plugin where ERROR occurred
    const char *mes;         ERROR message
} sceHiErrStateType;
```

Description

This is a structure for returning information about an ERROR that occurred within a plugin.

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

Library	*Introduced*	*Documentation last modified*
libhig	2.1	January 4, 2001

Structure

```
typedef struct _hig_head_data_t {
    char plug_name[12];                 Plugin block-specific name
    struct _sceHiPlug *plug_blk_addr;   Pointer to plugin block
} sceHiHeadData;
```

Description

This is the plugin registration portion of the data format header information.

It indicates the type of the plugin block to be used.

sceHiHeader

Header

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

Structure

typedef struct _sceHiList {

sceHiType *type;*	Identification code
u_int **addr;*	"Relative/absolute" pointer to inserted plugin block/data block
u_int *reserve;*	Reserved area for future expansion

} sceHiList;

Description

This is a structure for enumerating the inserted plugin blocks and data blocks.

addr is initially a relative address. Parsing is performed to rewrite it as an absolute address.

sceHiPlug

Plugin block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

Structure

typedef struct _sceHiPlug {

sceHiType *type;*	Type attribute of plugin
void **myapi;*	Pointer to plugin function
u_int *size;*	Size (qword size) of plugin block
char *nplug;*	Number of inserted plugin blocks that have been set in the plugin block
char *ndata;*	Number of data blocks that have been set in the plugin block
char *reserve[6];*	Reserved area for future expansion
u_int *stack;*	Stack location of management information to be used by plugin
u_int *args;*	Location for data communication between application program and plugin
qword *list;*	Inserted plugin block list or data block list

} sceHiPlug;

Description

This is the plugin block type.

For a virtual plugin, the myapi member will contain NULL.

The args member acts as an interface for exchanging information between the user and plugin function.

Arguments are passed to the plugin by entering values in the args member and exchanging the information.

sceHiPlugTable

Plugin table

Library	*Introduced*	*Documentation last modified*
libhig	2.1	January 4, 2001

Structure

```
typedef struct _sceHiPlugTable {
    sceHiType type;          Type attribute of plugin
    void *func;              Pointer to plugin function
} sceHiPlugTable;
```

Description

This is a plugin registration structure that becomes an argument of sceHiRegistTable().

The type information of a plugin is associated in a one-to-one fashion with a function pointer. Plugins which are not required should be deleted from this table.

sceHiType

Type information

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

Structure

typedef struct _sceHiType {

unsigned long *repository:8;*	Plugin and data repository identifier (constant value)
unsigned long *project:8;*	Plugin and data project identifier (constant value)
unsigned long *category:8;*	Plugin and data category identifier (constant value)
unsigned long *status:8;*	Plugin and data status information (bit packed)
unsigned long *id:24;*	Plugin and data identifier
unsigned long *revision:8;*	Plugin and data revision number

} sceHiType;

Description

An application programmer specifies a plugin and data for the library by creating this sceHiType structure and passing it as an argument.

Functions

sceHiAddDataBlk

Add data block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiAddDataBlk(

sceHiPlug **plug;*	Destination plugin block
sceHiData **data;***)**	Source data block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function adds a data block to a plugin block by finding an empty list location.

If no empty location is found, an error is returned.

Return value

sceHiErr type

sceHiAddPlugBlk

Add plugin block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiAddPlugBlk(

sceHiPlug **plug1;*	Destination plugin block
sceHiPlug **plug2;***)**	Source data block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function adds the source plugin block to the destination plugin block by finding an empty list location.

If no empty location is found, an error is returned.

Return value

sceHiErr type

sceHiCallPlug

Call plugin function

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiCallPlug(

sceHiPlug **plug;*	Plugin process identifier
int *process;***)**	Source data block

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function processes the plugin block indicated by plug according to the value specified by process.

If the plugin block contains inserted plugin blocks, the function is called recursively.

Return value

sceHiErr type

sceHiContPlugListStatus

Resume plug-in function activation

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

sceHiErr type

sceHiContPlugStatus

Resume plug-in function activation

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

sceHiErr type

sceHiGetData

Get data block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiGetData(

sceHiPlug **plug;*	Plugin block address
u_int ***data;*	Address of variable for receiving RAW data address
sceHiType *type;***)**	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

Description

This function searches the data blocks that are held by the plugin 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

sceHiErr type

sceHiGetDataPlace

Get data block list number

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiGetDataPlace(

sceHiPlug **plug,*	Address of plugin 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

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

sceHiErr type

sceHiGetInsPlug

Get inserted plug-in block

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

sceHiErr type

sceHiGetList

Get block list

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiGetList(

sceHiPlug **plug;*	Plugin block address
sceHiList ***list;*	Address of variable for receiving the list item address
sceHiType *type***);**	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

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 plugin block list and data block list maintained by the plugin block.

Return value

sceHiErr type

sceHiGetPlug

Get plugin block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiGetPlug(

u_int **data;*	Data format header address
char **name;*	Plugin block-specific name
sceHiPlug ***plug;***)**	Address of variable for receiving the plugin block address

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function stores in plug the address of the plugin block having the specified name, from among the plugins that are registered in the data format header at the address specified by data.

The plugin block is needed in order to start up a plugin function using sceHiCallPlug(), which is described later.

Return value

sceHiErr type

sceHiGetPlugList

Get plug-in block list

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

sceHiErr type

sceHiGetPlugPlace

Get plugin block list number

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiGetPlugPlace(

sceHiPlug **plug1,*	Address of plugin block to check
sceHiPlug **plug2,*	Address of plugin block to check
int **ofs***)**	List number of plug2

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function returns in ofs the list number position at which the plugin block specified by plug2 exists in the plugin list of the plugin block specified by plug1.

The ofs value that is returned is a list number that can be handled by a function such as sceHiInsPlugBlk().

If the plugin block specified by plug2 does not exist in the plugin list specified by plug1, an error is returned.

Return value

sceHiErr type

sceHiGetType

Get type information

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

Syntax

sceHiErr sceHiGetType(

sceHiType **type;*	Type information of copy source
sceHiType **htype***;)**	Type information of copy destination

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function copies type information.

For information related to bit images, refer to the type information for high-level graphics library data formats.

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

sceHiErr type

sceHiInsDataBlk

Insert data block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiInsDataBlk(

sceHiPlug **plug;*	Destination plugin 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

Description

This function inserts the data block at the specified list number location in the plugin block.

If the specified list number location is not empty, an error is returned.

Return value

sceHiErr type

sceHiInsPlugBlk

Insert plugin block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiInsPlugBlk(

sceHiPlug **plug1;*	Destination plugin block
sceHiPlug **plug2;*	Source plugin block
int *nb;***)**	List number

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function inserts the specified plugin block at the specified list number location in the plugin block.

If the specified list number location is not empty, an error is returned.

Return value

sceHiErr type

sceHiMakeDataBlk

Create data block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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.

Return value

sceHiErr type

sceHiMakeType

Generate type information

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

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

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

sceHiErr type

sceHiNewPlugBlk

Create plugin block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiNewPlugBlk(

int *nplug;*	Maximum number of plugins to be registered
int *ndata;*	Maximum number of sets of data to be registered
sceHiPlug ***plug;*	Address of empty plugin block that will be created
sceHiType **type)*	Type of plugin to be registered

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function creates an empty plugin block

Memory is allocated internally from the HIG heap.

To destroy the created plugin block, use sceHiMemFree().

Return value

sceHiErr type

sceHiParseHeader

Analyze data format

Library	*Introduced*	*Documentation last modified*
libhig	2.1	July 2, 2001

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

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 plugin 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

sceHiErr type

sceHiRegistTable

Register plugin function

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiRegistTable(

sceHiPlugTable **table;*	Plugin registration table
u_int *num***);**	Number of tables

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function uses the plugin registration table to register a plugin function.

A plugin function that has not been registered cannot be used.

Return value

sceHiErr type

sceHiRmvDataBlk

Remove data block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiRmvDataBlk(

sceHiPlug **plug;*	Plugin 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

Description

This function deletes the list information of the specified data block from the plugin 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

sceHiErr type

sceHiRmvPlugBlk

Remove plugin block

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiRmvPlugBlk(

sceHiPlug **plug1;*	Plugin block from which plugin block registration is to be deleted
sceHiPlug **plug2;***)**	Target plugin 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

Description

This function deletes the list information of the specified plugin block from the plugin block.

The specified plugin block itself is not deleted.

The deleted list location is assumed to be empty, and a plugin block can be added or inserted at that location.

Return value

sceHiErr type

sceHiSetDataType

Set data block type attribute

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiSetDataType(

sceHiData **data;*	Data 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

Description

This function sets the type attribute for the specified data block.

Return value

sceHiErr type

sceHiSetPluginApi

Set plugin function

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiSetPluginApi(

sceHiPlug **plug;*) Plugin block for which plugin function is to be set

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function registers a plugin function for a plugin block.

The plugin function will be the function that was specified with the type attribute.

Return value

sceHiErr type

sceHiSetPlugType

Set plugin block type attribute

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiSetPlugType(

sceHiPlug **plug;*	Plugin 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

Description

This function sets the type attribute for the plugin block that was created with sceHiNewPlugBlk().

Return value

sceHiErr type

sceHiStopPlugListStatus

Inhibit plug-in function activation

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

sceHiErr type

sceHiStopPlugStatus

Inhibit plug-in function activation

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

sceHiErr type

Chapter 5: GS Register Management Services
Table of Contents

Common Structures

sceHiGsGiftag

Giftag structure

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Structure

typedef struct _sceHiGsMemTbl{

u_int *align;*	Alignment size (in words) Either of the following: SCE_HIGS_PAGE_ALIGN = 2048 SCE_HIGS_BLOCK = 64
u_int *addr;*	Address of the reserved region (in words)
u_int *size*	Size of the reserved region (in words)
struct _sceHiGsMemTbl **next;*	Pointer to the next GS memory table Constitutes the chunk table

} sceHiGsMemTbl;

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

Library	*Introduced*	*Documentation last modified*
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:	SCE_HIGS_VALID_FRAME
To transfer the zbuf register:	SCE_HIGS_VALID_ZBUF
To transfer the tex0 register:	SCE_HIGS_VALID_TEX0
To transfer the tex1 register:	SCE_HIGS_VALID_TEX1
To transfer the tex2 register:	SCE_HIGS_VALID_TEX2
To transfer the miptbp1 register:	SCE_HIGS_VALID_MIPTBP1
To transfer the miptbp2 register:	SCE_HIGS_VALID_MIPTBP2
To transfer the clamp register:	SCE_HIGS_VALID_CLAMP
To transfer the test register:	SCE_HIGS_VALID_TEST
To transfer the alpha register:	SCE_HIGS_VALID_ALPHA
To transfer the xyoffset register:	SCE_HIGS_VALID_XYOFFSET
To transfer the scissor register:	SCE_HIGS_VALID_SCISSOR
To transfer the fba register:	SCE_HIGS_VALID_FBA

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

Library	*Introduced*	*Documentation last modified*
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 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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

sceHiErr sceHiGsDisplayEnd(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Frees GS memory reserved by sceHiGsDisplaySet.

Internally calls sceHiGsMemFree.

Return value

SCE_HIG_NO_ERR Processing was successful

Error returned by sceHiGsMemFree

sceHiGsDisplayMode

Set display mode

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_VALUE	Mode specification incorrect

sceHiGsDisplaySet

Set display

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

sceHiErr sceHiGsDisplaySet(

u_int *w,*	Width
u_int *h,*	Height
u_int *psm,*	Frame buffer storage format
u_int *zpsm***)**	Depth buffer storage format

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Insufficient GS memory heap area

sceHiGsDisplaySize

Set display size

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

sceHiErr sceHiGsDisplaySize(

u_int *width*	Width
u_int *height***)**	Height

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_VALUE	Mode specification was incorrect

sceHiGsDisplayStatus

Display structure state

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

sceHiGsDisplay *sceHiGsDisplayStatus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns a pointer to the current display structure.

Return value

sceHiGsDisplay* Pointer to the display structure

sceHiGsDisplaySwap

Swap display buffers

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

GS Local Memory Management Functions

sceHiGsBlockSize

Acquire block size

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

size_t sceHiGsBlockSize(

u_int *w,*	Width
u_int *h,*	Height
u_int *psm***)**	Pixel storage format

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

The size as a function of block alignment (64word) is returned in units of word.

Return value

size_t	Size (in words)

sceHiGsMemAddTbl

Add GS memory area

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_VALUE	*tbl* is NULL

sceHiGsMemAlloc

Reserve GS memory area

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

sceHiGsMemTbl *sceHiGsMemAlloc(

u_int *align,*	Alignment of the area (in words)
size_t *size***)**	Size to be reserved (in words)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

An area with the specified alignment and of the specified size is reserved.

The table following the *next* member of the sceHiGsMemTbl is connected and a chunk table for use in management is constructed.

A search for an empty area within areas previously reserved is performed. If a gap exists then an insertion is performed, otherwise the area is appended at the end.

An area of the heap described by sizeof(sceHiGsMemTbl), is consumed.

Return value

Pointer to a sceHiGsMemTbl type

In case of failure, NULL is returned.

sceHiGsMemFree

Free GS memory area

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

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

Description

Frees the area of the argument *tbl*.

The structure of the chunk table is corrected.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_VALUE	*tbl* argument is NULL

sceHiGsMemInit

Initialize GS memory management

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

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

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_NO_HEAP	Overflow of the specified area
SCE_HIG_INVALID_VALUE	The size is 0

sceHiGsMemPrintTbl

Display GS memory area information

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

sceHiErr sceHiGsMemPrintTbl(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Displays information for all tables for which areas have been reserved.

All the values are displayed in hexadecimal notation.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsMemRealloc

Reallocate GS memory area

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

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

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

Pointer to sceHiGsMemTbl type

Returns NULL on failure or if the arguments are incorrect

sceHiGsMemRestSize

Residual size of GS memory area

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

size_t sceHiGsMemRestSize(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns the remaining size starting from the last address of the reserved area.

Return value

size_t Remaining size (in words)

sceHiGsMemRestSizePlus

Available space in GS memory area

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

size_t sceHiGsMemRestSizePlus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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 Available space (in words)

sceHiGsPageSize

Acquire page size

Library	*Introduced*	*Documentation last modified*
libhig	2.2	March 26, 2001

Syntax

size_t sceHiGsPageSize(

u_int *w,*	Width
u_int *h,*	Height
u_int *psm***)**	Pixel storage format

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

Returns the size as a function of the page alignment (2Kword) in words.

Return value

size_t	Size (in words)

GS Register Setting Functions

sceHiGsCtxChkSize

Calculate amount of GS memory used

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

GS memory usage size (units: words)

sceHiGsCtxCopy

Copy port (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxCopy(

sceHiGsCtx **dst,*	Pointer to management structure of destination port
sceHiGsCtx **src***)**	Pointer to management structure of source port

Calling conditions

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxCreate

Create port (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiGsCtx *sceHiGsCtxCreate(

int *isDbuf***)** Double buffer flag

Calling conditions

Can be called from a thread

Not multithread safe

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 succeeds, a pointer to the management structure of the created context register group is returned.

When processing fails, a NULL pointer is returned.

sceHiGsCtxDelete

Free port (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxDelete(

sceHiGsCtx **gsctx*) Pointer to management structure of port to be freed

Calling conditions

None

Description

This function frees the memory used by the port.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_DATA	Argument port is invalid

sceHiGsCtxFcache

Flush transfer area cache (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxFcache(

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

Cannot be called from an interrupt handler

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

sceHiGsCtxFcachel() should be used within an interrupt handler.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxFcachel

Flush transfer area cache (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxFcachel (

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsCtxGetDefault

Get default port (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiGsCtx *sceHiGsCtxGetDefault(void)

Calling conditions

None

Description

This function gets the default port that was set by the sceHiGsCtxSetDefault() function.

Return value

Pointer to management structure of context register group's default port

sceHiGsCtxGetRect

Get screen size

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxGetTex0

Get tex0 register value for using port as texture

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsCtxRegist

Register transfer (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	Transfer register function returned a non-zero value

sceHiGsCtxRegistClear

Register transfer for clear transfer area only (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

Syntax

sceHiErr sceHiGsCtxRegistClear(

sceHiGsCtx **gsctx*) — Pointer to management structure of port for which transfer is to be registered

Calling conditions

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	Transfer register function returned a non-zero value

sceHiGsCtxSend

Transfer immediately (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	October 11, 2001

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 sceHiGsCtxRegist() function.

Return value

SCE_HIG_NO_ERR Processing was successful

sceHiGsCtxSendClear

Transfer clear transfer area immediately (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsCtxSetByDBuff

Set port by using libgraph double buffer setting

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSetByDBuff(

sceHiGsCtx **gsctx,*	Pointer to management structure of port to be set
sceGsDBuff **dbuf***)**	Pointer to libgraph double buffer setting

Calling conditions

None

Description

This function uses the libgraph double buffer setting to conveniently set the context register group port. However, it will not completely reflect the original setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetClearColor

Set clear color

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetClearMode

Set clear mode

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetClearZ

Set clear depth

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets the Z-value of the sprite to be used for clearing.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetColorDepth

Set frame buffer format

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetContext

Switch context

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_VALUE	Invalid context ID

sceHiGsCtxSetDefault

Set default port (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR — Processing was successful

sceHiGsCtxSetDefaultValues

Set screen size and buffer format

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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, miptbp1, and miptbp2.

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetDepth

Set buffer format

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_DATA	isZbuf value is invalid

sceHiGsCtxSetEachBuffer

Allocate GS memory

Library	*Introduced*	*Documentation last modified*
libhig	2.3	October 11, 2001

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->validregs|=SCE_HIGS_VALID_FRAME|SCE_HIGS_VALID_ZBUF;
```

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetFrame

Set FRAME register and frame buffer pointers

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSetFrame(

sceHiGsCtx **gsctx,,*	Pointer to management structure of port for which values are to be set
int *fbp0,*	Buffer pointer to be set for management structure fbp[0]
int *fbp1,*	Buffer pointer to be set for management structure fbp[1]
int *fbw,*	FRAME register fbw
int *psm,*	FRAME register psm
int *fbmsk***)**	FRAME register fbmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetLumpBuffer

Allocate GS memory

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSetLumpBuffer(

sceHiGsCtx **gsctx,*	Pointer to management structure of port for which value is to be set
u_int *bp***)**	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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetMax

Change the maximum number of context register group ports

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSetMax(

int *num***)** Maximum value to be newly set

Calling conditions

None

Description

A context register group has a limit on the number of ports that can be obtained. The initial value is 256.

This function can change the maximum number as long as it is called before the sceHiGsServiceInit() function is called.

If this function is called after the sceHiGsServiceInit() function has already been called, it does nothing and an error is returned.

Notes

Each time the maximum value is increased by 1, the HiG memory area that is dynamically obtained by the sceHiGsServiceInit() function is increased by 4 bytes. Also, the processing performed when the sceHiGsCtxDelete() function is called becomes slightly slower linearly.

Return value

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	sceHiGsServiceInit() has already been called

sceHiGsCtxSetRect

Set screen size

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsCtxSetRegAlpha

Set ALPHA register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets a value in the port's register setting area, and enables the Alpha register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegClamp

Set CLAMP register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets a value in the port's register setting area and enables the Clamp register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegFba

Set FBA register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets a value in the port's register setting area and enables the Fba register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegMiptbp1

Set MIPTBP1 register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets a value in the port's register setting area and enables the Miptbp1 register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegMiptbp2

Set MIPTBP2 register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets a value in the port's register setting area and enables the Miptbp2 register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegTest

Set TEST register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets a value in the port's register setting area and enables the Test register's transfer setting.

Return value

SCE_HIG_NO_ERRProcessing was successful

sceHiGsCtxSetRegTex0

Set TEX0 register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSetRegTex0(

sceHiGsCtx **gsctx,,*	Pointer to management structure of port for which value is to be set
int *tbp0,*	TEX0 register tbp0
int *tbw,*	TEX0 register tbw
int *psm,*	TEX0 register psm
int *tw,*	TEX0 register tw
int *th,*	TEX0 register th
int *tcc,*	TEX0 register tcc
int *tfx,*	TEX0 register tfx
int *cbp,*	TEX0 register cbp
int *cpsm,*	TEX0 register cpsm
int *csm,*	TEX0 register csm
int *csa,*	TEX0 register csa
int *cld***)**	TEX0 register cld

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function sets a value in the port's register setting area and enables the Tex0 register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegTex1

Set TEX1 register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

Description

This function sets a value in the port's register setting area and enables the Tex1 register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegXyoffset

Set XYOFFSET register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSetRegXyoffset(

sceHiGsCtx **gsctx,,*	Pointer to management structure of port for which value is to be set
int *ofx,*	XYOFFSET register ofx
	XYOFFSET register ofy
int *ofy*)	

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function sets a value in the port's register setting area and enables the Xyoffset register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetRegZbuf

Set ZBUF register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSetRegZbuf(

sceHiGsCtx **gsctx,,*	Pointer to management structure of port for which value is to be set
int *zbp,*	ZBUF register fbp
int *psm,*	ZBUF register psm
int *zmsk***)**	ZBUF register zmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function sets a value in the port's register setting area and enables the Zbuf register's transfer setting.

Return value

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSetZbufDepth

Set Z-buffer format

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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, SCE_GS_PSMCT16, 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->validregs|=SCE_HIGS_VALID_ZBUF|SCE_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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_DATA	isZbuf value is invalid

sceHiGsCtxShiftBuffers

Shift GS memory assignment

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSwap

Swap double buffer and set interlace

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxSwapAll

Swap double buffer and set interlace (for all synchronous ports of the context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxSwapAll(

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 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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsCtxUpdate

Update transfer area (context register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsCtxUpdate(

sceHiGsCtx **gsctx*) — Pointer to management structure of port for which value is to be updated

Calling conditions

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsEnvCopy

Copy port (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsEnvCopy(

sceHiGsEnv **dst,*	Pointer to management structure of destination port
sceHiGsEnv **src***)**	Pointer to management structure of source port

Calling conditions

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	An attempt was made to copy a port having a different transfer register setting

sceHiGsEnvCreate

Create port (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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 succeeds, a pointer to the management structure of the created port is returned.

When processing fails, a NULL pointer is returned.

sceHiGsEnvDelete

Free port (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsEnvDelete(

sceHiGsEnv **gsenv*) — Pointer to management structure of port to be released

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function frees the memory used by the port.

Return value

sceHiErr type

sceHiGsEnvFcache

Flush transfer area cache (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

sceHiGsEnvFcacheI() should be used within the interrupt handler.

Return value

sceHiErr type

sceHiGsEnvFcachel

Flush transfer area cache (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsEnvFcachel(

sceHiGsEnv **gsenv*) — Pointer to management structure of port for which cache is to be flushed

Calling conditions

Can be called from interrupt handler

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

sceHiErr type

sceHiGsEnvGetDefault

Get default port (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiGsEnv *sceHiGsEnvGetDefault(void)

Calling conditions

None

Description

This function gets the default port that was set with the sceHiGsEnvSetDefault() function.

Return value

Pointer to management structure of environment register group's default port

sceHiGsEnvRegist

Register transfer (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsEnvRegist(

sceHiGsEnv **gsenv*) — Pointer to management structure of port for which transfer is to be registered

Calling conditions

Can be called from a thread

Not multithread safe

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

sceHiErr type

sceHiGsEnvSend

Transfer immediately (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

sceHiErr type

sceHiGsEnvSetDefault

Set default port (environment register group)

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

sceHiErr type

sceHiGsEnvSetRegBitblt

Set Bitblt register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvSetRegColclamp

Set COLCLAMP register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegDimx

Set DIMX register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegDthe

Set DTHE register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegFog

Set FOG register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegFogcol

Set FOGCOL register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegLabel

Set Label register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvSetRegPabe

Set PABE register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegPrmode

Set PRMODE register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegPrmodecont

Set PRMODECONT register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegScanmsk

Set Scanmsk register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvSetRegSignal

Set Signal register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvSetRegTexa

Set TEXA register

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	This register's transfer setting was disabled

sceHiGsEnvSetRegTexclut

Set Texclut register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvSetRegTrxdir

Set Trxdir register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvSetRegTrxpos

Set Trxpos register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvSetRegTrxreg

Set Trxreg register

Library	*Introduced*	*Documentation last modified*
libhig	2.4	October 11, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	The register transfer setting was invalid

sceHiGsEnvUpdate

Update transfer area of environment register group port

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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

sceHiErr type

sceHiGsServiceExit

Exit GS register management service

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsServiceExit(void)

Calling conditions

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsServiceInit

Initialize GS register management service

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

Syntax

sceHiErr sceHiGsServiceInit(void)

Calling conditions

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_FAILURE	sceHiGsServiceInit() has already been called

sceHiGsServiceSetRegistFunc

Set function to be used for transfer registration

Library	*Introduced*	*Documentation last modified*
libhig	2.3	July 2, 2001

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.

Return value

SCE_HIG_NO_ERR Processing was successful

Old GS Register Setting Functions

sceHiGsAlphaRegs

Set ALPHA register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsAlphaRegs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsClampRegs

Set CLAMP register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsClampRegs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsClear

Set clear

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsClearColor

Set clear color

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsClearColor(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsClearDepth

Set clear depth

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsClearDepth(

u_int *z***,)** Z-value

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsColclampRegs

Set COLCLAMP register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsContextID

Switch context

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsContextID(

int *id***)** Context ID

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_VALUE	Context ID is invalid

sceHiGsContextStatus

Get current context

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiGsContext *sceHiGsContextStatus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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

sceHiGsContext pointer

sceHiGsDimxRegs

Set DIMX register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsDimxRegs(

int *dm***[16])** DIMX register dm

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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:

sceHiGsCtxSetRegDimx (sceHiGsStdCtx, dm);

Return value

SCE_HIG_NO_ERR Processing was successful

sceHiGsDtheRegs

Set DTHE register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsDtheRegs(

int *dthe***)** DTHE register dthe

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsFbaRegs

Set FBA register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsFogcolRegs

Set FOGCOL register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsFogcolRegs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsFogRegs

Set FOG register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsFrameRegs

Set FRAME register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsFrameRegs(

int *fbp,*	FRAME register fbp
int *fbw,*	FRAME register fbw
int *psm,*	FRAME register psm
int *fbmsk***)**	FRAME register fbmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsGeneralStatus

Get general-purpose register contents

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiGsGeneral *sceHiGsGeneralStatus(void)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

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

sceHiGsGeneral pointer

sceHiGsMiptbp1Regs

Set MIPTBP1 register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsMiptbp1Regs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsMiptbp2Regs

Set MIPTBP2 register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsMiptbp2Regs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsPabeRegs

Set PABE register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsPackedCreate

Create PACKED data

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiGsPacked *sceHiGsPackedCreate(

u_char **addr,*	Output destination GS register array
u_short *n***)**	Number of register arrays

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

This function creates PACKED data.

It creates a suitable giftag and a PATH2 transfer packet.

The packet chain id is entered in the id of the giftag member.

sceHiDMAMake_ChainStart, sceHiDMAMake_WaitMicro, sceHiDMAMake_LoadGS, and sceHiDMAMake_ChainEnd are called internally.

The memory for the giftag and packed members is made contiguous.

sizeof(sceHiGsPacked), sizeof(sceHiGsGiftag), and sizeof(sceHiGsPacked_t)*n heap areas are used and an n+2 qword packet size is also used.

This function is retained for compatibility with previous libraries.

You should use the GS register management service function instead.

Return value

sceHiGsPacked* sceHiGsPacked-type pointer

When processing fails, NULL is returned.

sceHiGsPackedDelete

Delete PACKED data

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

Error that is returned by sceHiDMADel_Chain

sceHiGsPackedUpdate

Update PACKED data

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful
SCE_HIG_INVALID_VALUE	p is NULL

sceHiGsPrmodecontRegs

Set PRMODECONT register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR Processing was successful

sceHiGsPrmodeRegs

Set PRMODE register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsPrmodeRegs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsTestRegs

Set TEST register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsTestRegs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsTex0Regs

Set TEX0 register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsTex0Regs(

int *tbp0,*	TEX0 register tbp0
int *tbw,*	TEX0 register tbw
int *psm,*	TEX0 register psm
int *tw,*	TEX0 register tw
int *th,*	TEX0 register th
int *tcc,*	TEX0 register tcc
int *tfx,*	TEX0 register tfx
int *cbp,*	TEX0 register cbp
int *cpsm,*	TEX0 register cpsm
int *csm,*	TEX0 register csm
int *csa,*	TEX0 register csa
int *cld***)**	TEX0 register cld

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsTex1Regs

Set TEX1 register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsTex1Regs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsTexaRegs

Set TEXA register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsTexaRegs(

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsXyoffsetRegs

Set XYOFFSET register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

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

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

SCE_HIG_NO_ERR	Processing was successful

sceHiGsZbufRegs

Set ZBUF register

Library	*Introduced*	*Documentation last modified*
libhig	2.2	July 2, 2001

Syntax

sceHiErr sceHiGsZbufRegs(

int *zbp,*	ZBUF register fbp
int *psm,*	ZBUF register psm
int *zmsk***)**	ZBUF register zmsk

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

SCE_HIG_NO_ERR	Processing was successful

Chapter 6: Memory Area Management Services
Table of Contents

Functions

sceHiMemAlign

Allocate memory with alignment

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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).

If allocation fails, NULL is returned.

Return value

Pointer to allocated area

sceHiMemAlloc

Allocate memory

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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.

If allocation fails, NULL is returned.

Return value

Pointer to allocated area

sceHiMemCalloc

Allocate memory with initialization

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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.

If allocation fails, NULL is returned.

Return value

Pointer to allocated area

sceHiMemFree

Free memory

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

void sceHiMemFree(
void **p;*) Pointer

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

sceHiMemGetNousedSize

Get unused heap size

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiMemGetNousedsize(

int **size*) Address of variable for receiving remaining capacity

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Multithread safe

Description

This function returns in the *size* argument the remaining capacity of the heap that is managed by the library, in bytes.

Return value

sceHiErr type

sceHiMemGetUsedSize

Get used heap size

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

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

sceHiErr type

sceHiMemInit

Initialize heap area

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

Syntax

sceHiErr sceHiMemInit(

void **ptr;*	User allocated buffer pointer
size_t *size;***)**	User allocated buffer size

Calling conditions

Can be called from a thread

Not multithread safe

Description

This function sets the heap area that will be managed by the library to the specified size starting at the area specified by *ptr*.

The library and plugins dynamically use only memory within this area.

The user-side heap area is not destroyed.

This function should be called only once by an application that uses HiG.

Return value

sceHiErr type

sceHiMemRealloc

Reallocate memory

Library	*Introduced*	*Documentation last modified*
libhig	2.1	March 26, 2001

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

Description

This function reallocates the allocated area according to the specified size and returns a pointer to the allocated area.

If allocation fails, NULL is returned.

Return value

Pointer to allocated area

Chapter 7: High Level Graphics Plugin Library
Table of Contents

Structures

sceHiPlugAnimePreCalcArg_t

Animation plugin argument structure

Library	*Introduced*	*Documentation last modified*
libhip	2.2	July 2, 2001

Structure

typdef struct {

u_int *setframe;*	Frame number passed to animation plugin
int *setframe_enable;*	1: Pass setframe to plugin 0: Do not pass
u_int *currentframe;*	Current frame number returned by animation plugin

} **sceHiPlugAnimePreCalcArg_t;**

Description

This is an argument type passed to the SCE_HIG_PRE_PROCESS of the animation 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.

sceHiPlugClutBumpPreArg_t

ClutBump plug-in argument structure

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
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.

sceHiPlugHrchyPreCalcArg_t

Hierarchy plugin argument structure

Library	*Introduced*	*Documentation last modified*
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 hierarchy 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

Library	*Introduced*	*Documentation last modified*
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

Microcode attribute constant

Library	*Introduced*	*Documentation last modified*
libhip	2.2	March 26, 2001

Structure

```
typdef 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 microcode. Each attribute value is handled as a bit value and can be duplicated and held. Use "bit or" in a microcode attribute specification that has multiple attributes.

Example: Microcode attribute that handles Fog and performs Anti.

SCE_HIP_MICRO_ATTR_FGE | SCE_HIP_MICRO_ATTR_ANTI

Table 7-1

Constant	Meaning
SCE_HIP_MICRO_ATTR_NONE	No special attribute
SCE_HIP_MICRO_ATTR_FGE	Fog attribute
SCE_HIP_MICRO_ATTR_ANTI	Anti attribute

sceHiPlugMicroInitArg_t

Micro plugin argument structure

Library	*Introduced*	*Documentation last modified*
libhip	2.2	July 2, 2001

Structure

typedef struct {

sceHiPlugMicroTbl_t **tbl;*	Pointer to table that defines addresses and attributes of microcode
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 microcode is selected.

sceHiPlugMicroPreCalcArg_t

Micro plugin argument structure

Library	*Introduced*	*Documentation last modified*
libhip	2.2	July 2, 2001

Structure

typdef struct {

int *micro;*	micro code 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 micro code
float *fogbegin;*	Fog parameter passed to micro code
float *fogend;*	Fog parameter passed to micro code

} **sceHiPlugMicroPreCalcArg_t;**

Description

Argument type passed to the SCE_HIG_PRE_PROCESS of the 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.

The micro code is not transferred when -1 is given to the *micro* member. When using the *anticutoff, fogbegin* and *fogend* members, the corresponding microcodes must be registered.

sceHiPlugMicroTbl_t

Microcode registration table type

Library	*Introduced*	*Documentation last modified*
libhip	2.2	July 2, 2001

Structure

typdef struct {

u_int **micro;*	micro code pointer
u_int *attr;*	micro code attribute

} sceHiPlugMicroTbl_t

Description

This is a table type for registering microcode. The member *tbl* of argument type sceHiPlugMicroInitArg_t that micro plugins accept must be this type of array.

attr must by an OR constant defined in sceHiPlugMicroAttr_t.

sceHiPlugReflectPreArg_t

Reflection plug-in argument structure

Library	*Introduced*	*Documentation last modified*
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

Refraction plug-in argument structure

Library	*Introduced*	*Documentation last modified*
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
```

sceHiPlugShadowMapInitArg_t

ShadowMap plug-in argument structure

Library	*Introduced*	*Documentation last modified*
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.

sceHiPlugTex2dInitArg_t

Tex2D plugin argument structure

Library	*Introduced*	*Documentation last modified*
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.

sceHiPlugTim2InitArg_t

Tim2D plugin argument structure

Library	*Introduced*	*Documentation last modified*
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

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

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 7-2

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

Animation plugin

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

Syntax

sceHiErr sceHiPlugAnime(

sceHiPlug **plug;*	Pointer to animation 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

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 7-3

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 7-4:

Use	Amount Consumed
Internal buffer	1 (word/keyframe) + 1 (word/keyvalue) + 2 (qword)

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes

Table 7-5

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_ANIME_PLUG

Data Type Attributes

Table 7-6:

Repository	Project	Category	Data ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_ANIME	SCE_HIP_ANIME_DATA SCE_HIP_KEYFRAME SCE_HIP_KEYVALUE
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_HRCHY_DATA

Return value

sceHiErr type See HiG library reference.

sceHiPlugClutBump

ClutBump plug-in

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

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:

CLUT alpha = ambient alpha + diffuse alpha * (light.normal) + specular alpha *(light.normal)^shininess

Table 7-7

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 7-8

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword + 1 qword/clut

Plug-in Type Attribute

Table 7-9

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ BUMP	SCE_HIP_ CLUTBUMP_PLUG

Data Type Attribute

Repository	Project	Category	Data ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ BUMP	SCE_HIP_CLUTBUMP_D ATA
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ BUMP	SCE_HIP_ CLUTBUMP_NORMAL
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ SHAPE	SCE_HIP_ BASEMATRIX
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ TEX2D	SCE_HIP_ TEX2D_DATA

Return value

sceHiErr type See HiG library reference

sceHiPlugFishEye

FishEye plug-in

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

Description

This function performs scene rendering with a fish eye lens effect. To perform scene rendering, a separate microcode 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 7-10

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 7-11

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-12

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ REFLECT	SCE_HIP_ FISHEYE_PLUG

Return value

sceHiErr type	See HiG library reference

sceHiPlugHrchy

Hierarchy plugin

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

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

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 7-13:

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 7-14

Use	Amount Consumed
Internal buffer	1 (qword)

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes

Table 7-15

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_HRCHY_PLUG

Data Type Attributes

Table 7-16

Repository	Project	Category	Data ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_HRCHY	SCE_HIP_HRCHY_DATA SCE_HIP_PIVOT_DATA
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_BASEMATRIX

Return value

sceHiErr type See HiG library reference.

sceHiPlugLightMap

Light map plug-in

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

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 7-17

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 7-18

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-19

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ LIGHT	SCE_HIP_ LIGHTMAP_PLUG

Return value

sceHiErr type See HiG library reference

sceHiPlugMicro

Microcode plugin

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

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

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 7-20

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Reserves internal buffers. Generates DMA packets of installed microcode. Sets registration and attributes of multiple microcodes using sceHiPlugMicroInitArg_t.
SCE_HIG_PRE_PROCESS	Switches microcode using sceHiPlugMicroPreCalcArg_t in an actual plugin block argument (HiPlug type member args). If args is NULL, uses the microcode that only handles parallel light source maintained by the library.
SCE_HIG_POST_PROCESS	Registers a Chain using sceHiDMARegist to perform a DMA transfer of DMA packets of microcode. Does not perform the DMA transfer itself. Call sceHiDMASend(); 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 7-21

Use	Amount Consumed
Packet management area	1 qword
DMA packet buffers	4 qword
Internal buffers	1 qword

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes

Table 7-22

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_PLUG

Data Type Attributes

Table 7-23

Repository	Project	Category	Data ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_MICRO	SCE_HIP_MICRO_DATA

Return value

sceHiErr type See HiG library reference.

sceHiPlugReflect

Reflection plug-in

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

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 7-24

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 7-25

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_ REFLECT_PLUG

Return value

sceHiErr type See HiG library reference

sceHiPlugRefract

Refraction plug-in

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

Description

This function produces a refraction mapping effect.

Calculation expressions:

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)
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 7-27

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 7-28

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-29

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ REFLECT	SCE_HIP_ REFRACT_PLUG

Return value

sceHiErr type See HiG library reference

sceHiPlugShadowBox

Shadow box plug-in

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

Syntax

sceHiErr sceHiPlugShadowBox(

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

Description

The world coordinate values are calculated from the object's bounding box that will be shadowed. This bounding box effect is used by the ShadowMap plug-in.

World coordinate vertices are calculated from the hierarchical structure root matrix.

This plug-in's PRE_PROCESS must be called before the ShadowMap plug-in's PRE_PROCESS.

Table 7-30

Specification	Description
SCE_HIG_INIT_PROCESS	Allocate internal buffers.
SCE_HIG_PRE_PROCESS	Calculate world coordinate values of 8 vertices that define the bounding box.
SCE_HIG_POST_PROCESS	No operation.
SCE_HIG_END_PROCESS	Free internal buffers.

Amount of Memory Used

Table 7-31

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-32

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ SHADOW	SCE_HIP_ SHADOWBOX_PLUG

Data Type Attribute

Table 7-33

Repository	Project	Category	Data ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ SHADOW	SCE_HIP_ SHADOWBOX_DATA

Return value

sceHiErr type See HiG library reference

sceHiPlugShadowMap

Shadow map plug-in

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

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 7-34

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 7-35

Purpose	Amount Used
Packet management area	None
DMA packet buffer	None
Internal buffers	1 qword

Plug-in Type Attribute

Table 7-36

Repository	Project	Category	Plugin ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ SHADOW	SCE_HIP_ SHADOWMAP_PLUG

Data Type Attribute

Table 7-37

Repository	Project	Category	Data ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ SHADOW	SCE_HIP_ SHADOWBOX_DATA

Return value

sceHiErr type See HiG library reference

sceHiPlugShape

Shape plugin

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

Syntax

sceHiErr sceHiPlugShape(

sceHiPlug **plug;*	Pointer to shape 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

Description

This function draws shape data by creating packets for performing DMA transfers of the shape data to VU1 local memory, transferring the packets, and executing microcode.

The following processing is called according to the value specified by the process argument.

Table 7-38

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Allocates internal buffers. Analyzes shape data, matrix data, and 2D texture data and creates packets for performing DMA transfers of the packets to VU Mem1.
SCE_HIG_PRE_PROCESS	Performs no processing.
SCE_HIG_POST_PROCESS	Uses sceHiDMARegist to register chains for performing DMA transfers of shape data to VU Mem1. Does not perform actual DMA transfers. sceHiDMASend(); must be called separately.
SCE_HIG_END_PROCESS	Frees DMA packet buffers. Frees internal buffers.

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.

Table 7-39: Amount of Memory Consumed

Use	Amount Consumed
Packet management area	1 (qword/shape) + 1 (qword).
DMA packet buffers	2 (qword/matrix) + 1 qword (unnecessary if there is no matrix). 5 (qword/shape). 3 (qword/material) (unnecessary if there is no texture). 11 (qword /63vertices /geometry).
Internal buffers	3 (qword/shape) + 2 (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 7-40

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_SHAPE_PLUG

Data Type Attributes

Table 7-41

Repository	Project	Category	Data ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_SHAPE_DATA SCE_HIP_BASEMATRIX

Return value

sceHiErr type See HiG library reference.

sceHiPlugShapeInvisible

Set whether shape is visible or invisible

Library	*Introduced*	*Documentation last modified*
libhip	2.4	October 11, 2001

Syntax

sceHiErr sceHiPlugShapeInvisible(

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

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

sceHiErr type	See HiG library reference

SceHiPlugShapeMasterChainSetting

Change settings for Shape packet chain

Library	*Introduced*	*Documentation last modified*
libhip	2.4.2	December 3, 2001

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

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

sceHiErr type	See HiG library reference

sceHiPlugShare

Share plugin

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

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

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 7-42

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 7-43

Use	Amount Consumed
Internal buffers	2 (word/geometry)(in share) + 6 (qword)

The type attributes of this plugin and the type attributes of the required data are shown below.

Plugin Type Attributes

Table 7-44

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SHARE_PLUG

Data Type Attributes

Table 7-45

Repository	Project	Category	Data ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHARE	SCE_HIP_SHARE_DATA SCE_HIP_SRCDSTVERTEX SCE_HIP_SRCDSTNORMAL SCE_HIP_VERTEXINDEX SCE_HIP_NORMALINDEX SCE_HIP_SHAREVERTEX SCE_HIP_SHARENORMAL
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_SHAPE	SCE_HIP_SHAPE_DATA SCE_HIP_BASEMATRIX

Return value

sceHiErr type See HiG library reference.

sceHiPlugTex2D

2D texture plugin

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

Syntax

sceHiErr sceHiPlugTex2D(

sceHiPlug **plug;*	Pointer to 2D texture 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

Description

This function DMA transfers 2D texture data to GS local memory.

The following processing is called according to the value specified by the process argument.

Table 7-46:

Specified Value	Processing Description
SCE_HIG_INIT_PROCESS	Reserves internal buffers. Sets TBP and CBP of the GS transfer destination using sceHiPlugTex2dInitArg_t and specifies resident / non-resident Analyzes texture data. Creates DMA packets of texture data.
SCE_HIG_PRE_PROCESS	Performs no processing.
SCE_HIG_POST_PROCESS	Uses sceHiDMARegist(); to register texture data DMA packets for DMA transfer to the GS. Does not perform actual DMA transfers. sceHiDMASend(); must be called separately.
SCE_HIG_END_PROCESS	Frees DMA packet buffers. Frees internal buffers.

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 7-47

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 7-48

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TEX2D	SCE_HIP_TEX2D_PLUG

Data Type Attributes

Table 7-49

Repository	Project	Category	Data ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TEX2D	SCE_HIP_TEX2D_DATA

Return value

sceHiErr type See HiG library reference.

sceHiPlugTex2DSize

Get Tex2D plug-in texture size

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

Description

This function returns the total texture size which the Tex2D plug-in has.

Return value

size_t Texture size

sceHiPlugTim2

Tim2 texture plugin

Library	*Introduced*	*Documentation last modified*
libhip	2.1	July 2, 2001

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

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 7-50

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 sceHiPlugTim2InitArg_t 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 sceHiDMARegist();. Does not perform the DMA transfer itself. Call sceHiDMASend(); 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 7-51

Repository	Project	Category	Plugin ID
SCE_HIP_COMMON	SCE_HIP_FRAMEWORK	SCE_HIP_TIM2	SCE_HIP_TIM2PLUG

Data type attributes

Table 7-52

Repository	Project	Category	Data ID
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ TIM2	SCE_HIP_ TIM2_DATA
SCE_HIP_ COMMON	SCE_HIP_ FRAMEWORK	SCE_HIP_ TEX2D	SCE_HIP_ TEX2D_ENV

Return value

sceHiErr type (See HiG library reference)

sceHiPlugTim2GetName

Get Tim2 plug-in texture filename

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libhip	2.4	October 11, 2001

Syntax

sceHiErr sceHiPlugTim2GetNPictures (

sceHiPlug ***_plug_;**	Pointer to plugin block
int _n_**;**	Data block index
int ***_num_);**	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

Description

For the nth 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

sceHiErr type	See HiG library reference

sceHiPlugTim2Num

Get number of Tim2 plug-in texture files

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libhip	2.3	July 2, 2001

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

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

sceHiErr type	See HiG library reference

sceHiPlugTim2SetPicture

Set Tim2 plugin picture

Library	*Introduced*	*Documentation last modified*
libhip	2.4	October 11, 2001

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

Description

This function sets the data that *plug* will reference as the nth 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

sceHiErr type	See HiG library reference

Chapter 8: VU0 Library
Table of Contents

Structures

sceVu0FMATRIX

4x4 matrix

Library	*Introduced*	*Documentation last modified*
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)

Library	*Introduced*	*Documentation last modified*
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

sceVu0IVECTOR

4-dimensional vector (integer)

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	December 23, 1999

Structure

typedef int sceVu0IVECTOR[4];

Description

This is an (int X 4)-element vector. The EE is little endian, and the array elements of sceVu0IVECTOR 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

sceVu0IVECTOR 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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

Syntax

void sceVpu0Reset(void);

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Initializes VU0 and VIF0.

Return value

None

sceVu0AddVector

4-element parallel add (ADD/xyzw)

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Multiples 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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Copies matrix *m1* to matrix *m0*.

Return value

None

sceVu0CopyVector

Copy vector

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Copies vector *v1* to vector *v0*.

Return value

None

sceVu0CopyVectorXYZ

Copy vector

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Divides the vector *v1* by the scalar *q* and returns the result in vector *v0*.

Return value

None

sceVu0DivVectorXYZ

Divide

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Obtains the inner product of vectors *v0* and *v1*.

Return value

Inner product

sceVu0InterVector

Generate interpolation vector

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

Syntax

void sceVu0InterVectorXYZ(

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

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)

However, the element *w* of vector *v0* is returned directly in vector *v0*.

Return value

None

sceVu0InversMatrix

Generate inverse matrix

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Obtains 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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

Syntax

void sceVu0LightColorMatrix(

sceVu0FMATRIX *m,*	Output: Matrix
sceVu0FVECTOR *c0,*	Input: Vector (light source color 0)
sceVu0FVECTOR *c1,*	Input: Vector (light source color 1)
sceVu0FVECTOR *c2,*	Input: Vector (light source color 2)
sceVu0FVECTOR *a***);**	Input: Vector (ambient light color)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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)

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Normalizes vector *v1* and returns the result in *v0*.

Return value

None

sceVu0NormalLightMatrix

Generate normal light matrix

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

Syntax

void sceVu0NormalLightMatrix(

sceVu0FMATRIX *m,*	Output: Matrix
sceVu0FVECTOR *l0,*	Input: Vector (light source 0 direction)
sceVu0FVECTOR *l1,*	Input: Vector (light source 1 direction)
sceVu0FVECTOR *l2***)**	Input: Vector (light source 2 direction)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Obtains the outer product of vectors *v1* and *v2* and returns the result in *v0*.

Return value

None

sceVu0RotMatrix

Rotate matrix

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Peforms 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)

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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)

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	July 2, 2001

Syntax

void sceVu0ScaleVectorXYZ(

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

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)

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

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

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Translates matrix *m1* using vector *tv* and returns the result in *m0*.

Return value

None

sceVu0TransposeMatrix

Generate transposed matrix

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Obtains the transposed matrix of matrix *m1* and returns the result in *m0*.

Return value

None

sceVu0UnitMatrix

Generate unit matrix

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

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

Description

Returns a 4x4 unit matrix in matrix *m0*.

Return value

None

sceVu0ViewScreenMatrix

Generate view screen matrix

Library	*Introduced*	*Documentation last modified*
libvu0	1.1	March 26, 2001

Syntax

void sceVu0ViewScreenMatrix(

sceVu0FMATRIX *m,*	Output: Matrix
float *scrz,*	Input: (distance to screen)
float *ax,*	Input: (X-direction aspect ratio)
float *ay,*	Input: (Y-direction aspect ratio)
float *cx,*	Input: (X-coordinate of center of screen)
float *cy,*	Input: (Y-coordinate of center of screen)
float *zmin,*	Input: (Z-buffer minimum value)
float *zmax,*	Input: (Z-buffer maximum value)
float *nearz,*	Input: (Z of near clipping plane)
float *farz***)**	Input: (Z of far clipping plane)

Calling conditions

Can be called from an interrupt handler

Can be called from a thread

Not multithread safe

Description

Obtains the view screen matrix using the specified parameters and returns the result in *m*.

Return value

None