# Run-time Library Reference

Beta Release 3.6 - Dec 1996

# TABLE OF CONTENTS

# About This Manual

This manual is the reference for release 3.6 of the PlayStation OS Run-time Library. It defines all available library functions and structures. The companion Overview volume describes the structure and purpose of the libraries in programming games for the PlayStation.

# Changes Since Last Release

This manual includes a number of new functions and structures added since release 3.5. The changes in this Reference due to Run-time Library release 3.6 are described below.

**Note** that throughout this manual, an asterisk follows the names of functions and structures introduced at release 3.6.

## Kernel Library (libapi)

Functions Added

InitHeap2
malloc2
realloc2
calloc2
free2

## Basic Graphics Library (libgpu)

Function Added

IsIdleGPU

## Extended Graphics Library (libgs)

Functions Added

GsTMDfastTF4LM, GsTMDfastTF4LFGM, GsTMDfastTF4NLM, GsTMDfastTNF4M, GsTMDfastTG4LM, GsTMDfastTG4LFGM, GsTMDfastTG4NLM, GsTMDfastTNG4M, GsTMDdivTF4LM, GsTMDdivTF4LFGM, GsTMDdivTF4NLM, GsTMDdivTNF4M, GsTMDdivTG4LM, GsTMDdivTG4LFGM, GsTMDdivTG4NLM, GsTMDdivTNG4M, GsA4divTF4LM, GsA4divTF4LFGM, GsA4divTF4NLM, GsA4divTNF4M, GsA4divTG4LM, GsA4divTG4LFGM, GsA4divTG4NLM, GsA4divTNG4M

## Basic Geometry Library (libgte)

Functions Added

SetMulRotMatrix
MatrixNormal_0
CompMatrixLV

Function Description Updated

ApplyRotMatrixLV

## Data Processing Library (libpress)

Structure Added

ENCSPUENV

Function Added

EncSPU

## Extended Sound Library (libsnd)

Structure Added

_SsFCALL

Functions Added

dmy_Ss....
SsGetCurrentPoint
SsChannelMute
SsSeqOpenJ
SsSepOpenJ

## Basic Sound Library (libspu)

Functions Added

SpuSetEnv
SpuFlush
SpuNSetVoiceAttr
SpuSetVoiceVolume
SpuSetVoiceVolumeAttr
SpuSetVoicePitch
SpuSetVoiceNote
SpuSetVoiceSampleNote
SpuSetVoiceStartAddr
SpuSetVoiceLoopStartAddr
SpuSetVoiceAR
SpuSetVoiceDR
SpuSetVoiceSR
SpuSetVoiceRR
SpuSetVoiceSL
SpuSetVoiceARAttr
SpuSetVoiceSRAttr
SpuSetVoiceRRAttr
SpuSetVoiceADSR
SpuSetVoiceADSRAttr
SpuNGetVoiceAttr
SpuGetVoiceVolume
SpuGetVoiceVolumeAttr
SpuGetVoiceVolumeX
SpuGetVoicePitch
SpuGetVoiceNote
SpuGetVoiceSampleNote
SpuGetVoiceEnvelope
SpuGetVoiceStartAddr
SpuGetVoiceLoopStartAddr
SpuGetVoiceAR

SpuGetVoiceDR
SpuGetVoiceSR
SpuGetVoiceRR
SpuGetVoiceSL
SpuGetVoiceARAttr
SpuGetVoiceSRAttr
SpuGetVoiceRRAttr
SpuGetVoiceADSR
SpuGetVoiceADSRAttr
SpuGetVoiceEnvelopeAttr

### New Library

Serial Input/Output Library (libsio)

This is a newly available library from release 3.6. This is a library to perform standard I/O between PC and PS using the communication cable DTL-H3050. Since the standard I/O of the debugging station is set to NULL normally, no debug  information can be obtained. By using this library, libsio, standard I/O can be allocated to the PS communication port, and by connecting the communication cable DTL-H3050, RS232C I/O is enabled.

Functions Added

AddSIO
DelSIO
_sio_control

## Manual Structure

The Library Reference contains fourteen chapters providing definitions of library structures and functions.

Note that the Library Reference chapters are subject-oriented, rather than library-oriented. This means that you can expect fo find all related functions in a single chapter, regardless of the library in which they reside. For example, Chapter 10, the Controller/Peripherals Library, describes functions in libetc, libgun and libtap libraries.

The  Reference Stripe that appears at the top of each page describes:

- the related library
- the related header file
- the library release in which that function/structure was introduced
- the documentation version for that page

## Related Documentation

This manual should be read in conjunction with the Run-time Library Overview, since the Overview summarizes the use of the libraries.

The complete set of the Developer Reference Series includes the following:

- Programmer Board Set (DTL-H2000)
- PlayStation Operating System
- PlayStation Hardware
- Run-time Library Overview
- Run-time Library Reference
- Psy-Q Development System
- CD Emulator
- CD Generator

- 3D Graphics Tool
- Sprite Editor
- Sound Artist Tool
- File Formats

Note that the Developer Support BBS posts late-breaking developments regarding the run-time libraries and also provides notice of forthcoming documentation releases and upgrades.

## Typographical Conventions

Certain Typographic Conventions are used through out this manual to clarify the meaning of the text. The following details the specific conventions used to represent literals, arguments, keywords, etc.

The following conventions apply to all narrative text outside of the structure and function descriptions.

| Convention | Meaning |
|---|---|
| \| | A revision bar. Indicates that information to the left or right of the bar has been changed or added since the last release. |
| `courier` | Indicates literal program code. |
| **Bold** | Indicates a document, chapter or section title. |

The following conventions apply within structure and function descriptions only:

| Convention | Meaning |
|---|---|
| Medium Bold | Denotes structure or function types and names. |
| Italic | Denotes function arguments and structure members. |
| { } | Denotes the start and end of the member list in a structure declaration. |

## Ordering Information

To order printed copies of this or any other developer documentation, please contact Sony Computer Entertainment as follows:

In the USA:

Attn: 3rd Party Tools Coordinator
Sony Computer Entertainment America
919 East Hillsdale Blvd
Foster City, CA 94404

Tel (415) 655-8000

In Europe:

Attn: Production Coordinator
Sony Computer Entertainment Europe
Waverley House
7-12 Noel Street
London W1V 4HH

Tel: +44 (0) 171 447 1600

# Chapter 1: Kernel Library
# Table of Contents

## DIRENTRY

Data structure of directory entries.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Structure

**struct DIRENTRY {**
    **char** *name* [20];
    **long** *attr*;
    **long** *size*;
    **struct DIRENTRY** **next*
    **char** *system*[8];
**}**

### Members

| | |
|---|---|
| *name* | FIlename |
| *attr* | Attributes (dependent on file system) |
| *size* | FIle size (in bytes) |
| *next* | Pointer to next file entry (for user) |
| *system* | Reserved by system |

### Explanation

This structure stores information relating to files registered in the file system.

### Remarks

**See also:** firstfile (p. 1-24), nextfile (p. 1-44).

## EvCB

Event Control Block.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Structure

```
struct EvCB {
    unsigned long desc;
    long status;
    long spec;
    long mode;
    (long *FHandler)();
    long system[2];
};
```

### Members

| | |
|---|---|
| *desc* | Cause descriptor |
| *status* | Status |
| *spec* | Event type |
| *mode* | Mode |
| *FHandler* | Pointer to a function type handler |
| *system* | Reserved by system |

### Explanation

Used for event management.

### Remarks

**See also:** Open Event (p, 1-47), GetConf (p. 1-27), SetConf (p. 1-56).

## EXEC

The data structure of an execute file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Structure

```
struct EXEC {
    unsigned long pc0;
    unsigned long gp0;
    unsigned long t_addr;
    unsigned long t_size;
    unsigned long d_addr;
    unsigned long d_size;
    unsigned long b_addr;
    unsigned long b_size;
    unsigned long s_addr;
    unsigned long s_size;
    unsigned long sp;
    unsigned long fp;
    unsigned long gp;
    unsigned long ret;
    unsigned long base;
};
```

### Members

| | |
|---|---|
| *pc0* | Execution start address |
| *gp0* | gp register initial value |
| *t_addr* | Starting address of text section and initialized data section |
| *t_size* | Size of text section |
| *d_addr* | System reserved |
| *d_size* | System reserved |
| *b_addr* | Uninitialized data section start address |
| *b_size* | Uninitialized data section size |
| *s_addr* | Stack start address (specified by the user) |
| *s_size* | Stack size (specified by the user) |
| *sp* | Register shunt variable |
| *fp* | Register shunt variable |
| *gp* | Register shunt variable |
| *ret* | Register shunt variable |
| base | Register shunt variable |

### Explanation

Used by Exec() function.

### Remarks

**See also:** Exec (p. 1-22).

## TCB

Task Control Block.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Structure

**struct TCB {**
    **long** *status*;
    **long** *mode*;
    **unsigned long** *reg*[NREGS];
    **long** *system*[6];
**}**;

### Members

| | |
|---|---|
| *status* | Status |
| *mode* | Mode |
| *reg* | Register saving area (specified by register designation macro) |
| *system* | Reserved by system |

### Explanation

Data block where a context (the contents of the registers) is stored for thread management.

### Remarks

**See also:** Open Th (p. 1-48), ChangeTh (p. 1-11), GetConf (p. 1-27), SetConf (p. 1-56).

## TCBH

Task Execute Queue.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Structure

**struct TCBH {**
    **struct TCB** **entry*;
**}**;

### Members

| | |
|---|---|
| *entry* | Pointer to execute TCB. |

### Explanation

Used for thread management. The execute TCB is linked to *entry*.

### Remarks

**See also:** ChangeTh (p. 1-11).

## ToT

System Table Information.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Structure

**struct ToT {**
    **unsigned long** **head*;
    **long** *size*;
**}**;

### Members

| | |
|---|---|
| *head* | Pointer to a system table start address |
| *size* | System table size (in bytes) |

### Explanation

Table information which enables organized handling of various system tables which are used by the kernel. The placement address is 0x00000100.

### Remarks

## calloc2*

Allocates main memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.6* | *10/23/96* |

### Syntax

**#include** *<stdlib.h>*
**void** **calloc2( size_t n, size_t s )*

### Arguments

| | |
|---|---|
| *n* | Number of partitions |
| *s* | Size of one partition |

### Explanation

This function allocates a block of n*s bytes. Corresponds to InitHeap2().

### Return value

Returns a pointer to the allocated memory block. If allocation fails, NULL will be returned.

### Remarks

**See also:** malloc2(),realloc2(),free2()

## cd

Change default directory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long cd** (**path*)
**char** **path*;

### Arguments

| | |
|---|---|
| *path* | Pointer to the default directory path |

### Explanation

Changes the default directory path for the file system. The file system is specified by the device name at the beginning of the path.

### Return value

Returns “1” if it succeeds, and “0” otherwise.

### Remarks

**See also:**

## ChangeClearPAD

Sets the control driver.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void ChangeClearPAD** (*val*)
**long** *val*;

### Arguments

| | |
|---|---|
| *val* | Vertical retrace line interruption clear flag |

### Explanation

This function specifies whether to complete interrupt processing in a control driver started by a vertical retrace line interrupt, or to pass processing to a lower priority interrupt module without completion. A *val* value of 1 specifies completion, while a *val* value of 0 specifies passing.

### Return value

None.

### Remarks

**See also:** StartPAD (p. 1-60), StopPAD (p. 1-61), StartCARD (see libcard), StopCARD (see libcard).

## ChangeTh

Changing a thread to be executed.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long ChangeTh** (*thread*)
**unsigned long** *thread*;

### Arguments

| | |
|---|---|
| *thread* | Thread descriptor |

### Explanation

Execution is transferred to the thread specified by *thread*. The current thread is saved in a TCB during execution of this function. It returns from this function when the original thread is restored.

### Return value

On success and re-execution, the function returns 1. On failure, it returns 0. The Return value on re-execution can be changed by any other thread.

### Remarks

Before executing ChangeTh(), initialize TCB reg [R-SR] to the following:

- The interrupt context is 0X404
- The main flow is 0X401

**See also:** TCB structure (p.1-5), TCBH structure (p. 1-6), OpenTh (p. 1-48).

## CheckCallback

Determines whether the program is executing a callback.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *2.x* | *7/31/96* |

### Syntax

**int CheckCallback**()

### Arguments

None.

### Explanation

The CheckCallback() function determines whether the program is currently executing within a callback context or normal context.

### Return value

Normal context returns 0. Callback context returns 1.

### Remarks

**See also:**

## close

Closing a file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**int close** (*fd*)
**int** *fd*;

### Arguments

| | |
|---|---|
| *fd* | File descriptor |

### Explanation

This function closes a file descriptor.

### Return value

On success, the function returns *fd*. On failure, it returns -1.

### Remarks

**See also:** Open (p. 1-46).

## CloseEvent

Closing an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long CloseEvent** (*event*)
**unsigned long** *event*;

### Arguments

| | |
|---|---|
| *event* | Event descriptor |

### Explanation

Releases the EvCB specified by *event*.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

To be executed in a critical section.

**See also:** OpenEvent (p. 1-47), EnterCriticalSection (p. 1-21), SwEnterCriticalSection (p. 1-64).

## CloseTh

Closes a thread.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long closeTh** (*thread*)
**unsigned long** *thread*;

### Arguments

| | |
|---|---|
| *thread* | Thread descriptor |

### Explanation

This function closes a thread and releases its TCB.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

To be executed in a critical section.

**See also:** OpenTh (p. 1-48), EnterCriticalSection (p. 1-21), SwEnterCriticalSection (p. 1-64).

## delete

Deletes a file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long delete** (**name*)
**char** **name*;

### Arguments

| | |
|---|---|
| *name* | Pointer to a filename |

### Explanation

Deletes the file specified by *name*.

### Return value

Returns “1” if it succeeds, and “0” otherwise.

### Remarks

**See also:** undelete (p. 1-67).

## DeliverEvent

Generates an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void DeliverEvent** (*ev1*, *ev2*)
**unsigned long** *ev1*;
**long** *ev2*;

### Arguments

| | |
|---|---|
| *ev1* | Cause descriptor |
| *ev2* | Event class |

### Explanation

This function delivers an event if that event's current status is EvStACTIVE (event not yet generated, generation possible). If the event mode is EvMdNOINTR, the event handler function is called. If the event mode is EvMdINTR, the event status is changed to EvStALREADY (event already occurred, generation prohibited).

### Return value

None.

### Remarks

This function must be executed in a critical section.

**See also:** UnDeliverEvent (p. 1-68), OpenEvent (p. 1-47), TestEvent (p. 1-66), EnterCriticalSection (p. 1-21), SwEnterCriticalSection (p. 1-64), DisableEvent (p. 1-19), EnableEvent (p. 1-20) WaitEvent (p. 1-70), CloseEvent (p. 1-15).

## DisableEvent

Disables an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long DisableEvent** (*event*)
**unsigned long** *event*;

### Arguments

| | |
|---|---|
| *event* | Event descriptor |

### Explanation

This function inhibits occurrence of an event specified by the descriptor event. It changes the event status to EvStWAIT (event generation prohibited).

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

**See also:** EnableEvent (p. 1-19).

## EnableEvent

Enables occurrence of an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long EnableEvent** (*event*)
**unsigned long** *event*;

### Arguments

| | |
|---|---|
| *event* | Event descriptor |

### Explanation

This function enables occurrence of an event specified by the descriptor event. It changes the event status to EvStACTIVE (event not yet generated, generation possible).

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

**See also:** DisableEvent (p. 1-18), TestEvent (p. 1-66).

## EnterCriticalSection

Enter a critical section.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void EnterCriticalSection** (*void*)

### Arguments

None.

### Explanation

This function stops interrupts, and enters a critical section. This occurs immediately after kernel startup.

### Return value

None.

### Remarks

Executes an internal system call and destroys the interrupt context.

**See also:** TCBH (p. 1-6), TCB (p. 1-5), ExitCriticalSection (p. 1-23).

## Exception

Causes an interrupt.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void Exception** (*void*)

### Arguments

None.

### Explanation

This function causes an interrupt, and stores the current context in the execute TCB. It is also valid in a critical section.

### Return value

None.

### Remarks

Executes an internal call and destroys the exception context.

**See also:** TCBH (p. 1-6), TCB (p. 1-5), ChangeTh (p. 1-11), ReturnFromException (p. 1-54).

## Exec

Executes an execute file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long Exec** (**exec*, *argc*, **argv*)
**struct EXEC** **exec*;
**long** *argc*;
**char** **argv*[;

### Arguments

| | |
|---|---|
| *exec* | Pointer to execute file information |
| *argc* | Number of arguments |
| *argv* | Pointer to argument |

### Explanation

According to the execute file information specified by *exec*, this function executes a module already loaded in memory. If *exec*->s_addr is 0, neither stack or frame pointer is set.

The function performs the following:

- A data section without initial values is cleared to zero.
- sp, fp, and gp are saved, and then initialized. (fp is set to the same value as sp.)
- The arguments of main() are set (in the a0 and a1 registers).
- The execution start address is called.
- After a return is made, sp, fp, and gp are restored.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

To be executed in a critical section.

This function needs the ISO 9660 file system to run properly. _96_init() must be called to initialize this system and _96_remove must be called to exit this system.

**See also:** EXEC (p. 1-4), Load (p. 1-39), _96_init (p. 1-71), _96_remove (p. 1-72).

## ExitCriticalSection

Exits critical section.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void ExitCriticalSection** (*void*)

### Arguments

None.

### Explanation

This function enables interrupts, and exits from the critical section.

### Return value

None.

### Remarks

Executes an internal system call and destroys the interrupt context.

**See also:** TCBH (p. 1-6), TCB (p. 1-5), EnterCriticalSection (p. 1-20).

## firstfile

Looks up the first file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**struct DIRENTRY** ***firstfile** (**name*, **dir*)
**char** **name*;
**struct DIRENTRY** **dir*;

### Arguments

| | |
|---|---|
| *name* | Pointer to a filename |
| *dir* | Pointer to the buffer holding information relating to the referenced file. |

### Explanation

Looks up the file corresponding to the filename pattern *name*, and stores data relating to this file in the directory *dir*.

### Return value

Returns *dir* if it succeeds, and “0” otherwise.

### Remarks

The wildcard characters “?” (standing for any one character) and “*” (standing for a character string of any length) can be used in the filename pattern. Characters specified after “*” are ignored.

**See also:** DIRENTRY (p. 1-3), nextfile (p. 1-44).

## FlushCache

Flushes instruction cache (I cache).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void FlushCache** (*void*)

### Arguments

None.

### Explanation

Flushes I cache. Code is not executed when written to memory.

### Return value

None.

### Remarks

To be executed in a critical section.

## format

Initializes file system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long format** (**fs*)
**char** **fs*;

### Arguments

| | |
|---|---|
| *fs* | Pointer to file system name |

### Explanation

Initializes file system *fs*.

### Return value

Returns “1” if it succeeds, and “0” otherwise.

### Remarks

This function is only effective on writeable file systems.

**See also:**

## free2*

Frees allocated memory blocks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.6* | *10/23/96* |

### Syntax

**#include** *<stdlib.h>*
**void** *free2*
*(***void** **block)*

### Arguments

*<stdlib.h>*
**block*

### Explanation

This function releases a memory block that was allocated by calloc2,malloc2, and realloc2. Corresponds to InitHeap2().

### Return value

None.

### Remarks

**See also:** calloc2(),malloc2(),realloc2()

## GetConf

Obtains the kernel configuration.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**void GetConf** (**ev*, **tcb*, **sp*)
**unsigned long** **ev*;
**unsigned long** **tcb*;
**unsigned long** **sp*;

### Arguments

| | |
|---|---|
| *ev* | Pointer to the address that stores the number of event management block elements |
| *tcb* | Pointer to the address that stores the number of task management block elements |
| *sp* | Ignored |

### Explanation

This function stores a system configuration parameter set by SetConf () to the address given by the pointer as the argument.

### Return value

None.

### Remarks

This function returns an undefined value before the execution of SetConf () because this function refers to its internal parameter.

**See also:** SetConf (p. 1-56).

## GetCr

Gets a cause register value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long GetCr** (*void*)

### Arguments

None.

### Explanation

This function gets the control register cause register value.

The meaning of each bit of the cause register as follows:

**Table 1-1**

| Bit | Description |
|---|---|
| 31-6 | Reserved by the system |
| 5-2 | Exception code |
| | 0000 External interrupt |
| | 0001 Not used |
| | 0010 Not used |
| | 0011 Not used |
| | 0100 Address read error |
| | 0101 Address write error |
| | 0110 Command bus error |
| | 0111 Data bus error |
| | 1000 System call |
| | 1001 Break point |
| | 1010 Undefined command |
| | 1011 Co-processor not mounted |
| | 1100 Overflow |
| 1-0 | Reserved by the system |

### Return value

The current cause register value is returned.

### Remarks

**See also:** OpenTh (p. 1-48).

## GetGp

Gets a gp register value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long GetGp** (*void*)

### Arguments

None.

### Explanation

This function gets a gp register value.

### Return value

The current gp register value is returned.

### Remarks

**See also:** EXEC structure (p. 1-4), OpenTh (p. 1-48), Load (p. 1-39), Exec (p. 1-22).

## GetRCnt

Acquires a root counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long GetRCnt** (*spec*)
**long** *spec*;

### Arguments

| | |
|---|---|
| *spec* | Root counter |

### Explanation

Returns the current value of root counter *spec*. To be used when root counter spec has been set by SetRCnt to a polling mode (RCntMdNOINTR).

### Return value

On success, the function returns the 32-bit unsigned expanded counter value. On failure, it returns -1.

### Remarks

**See also:** SetRCnt (p. 1-58), StartRCnt (p. 1-61), StopRCnt (p. 1-63), ResetRCnt (p. 1-52).

## GetSp

Gets an sp register value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long GetSp** (*void*)

### Arguments

None.

### Explanation

This function gets an sp register value.

### Return value

A current sp register value is returned.

### Remarks

**See also:** EXEC (p. 1-4), OpenTh (p. 1-48), Load (p. 1-39), Exec (p. 1-22).

## GetSr

Gets a status register value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long GetSr** (*void*)

### Arguments

None.

### Explanation

This function gets the control register status register value.

The meaning of each bit of the status register is as follows:

**Table 1-2**

| Bit | Description |
|---|---|
| 31-28 | Co-processor installation flag (1: Installed) |
| | Bit 29 is GTE. |
| 27-11 | Reserved by the system |
| 10 | Always 1 |
| 9-3 | Reserved by the system |
| 2 | Main flow interrupt permission (1: Permission) |
| 1 | Reserved by the system |
| 0 | Interrupt permission (1: Permission) |

### Return value

The current status register value is returned.

### Remarks

**See also:** OpenTh (p. 1-48).

## GetSysSp

Gets a system stack.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**long GetSysSp** (*void*)

### Arguments

None.

### Explanation

This function acquires the highest address of a system stack area for event handler function execution.

The size of the stack area is 2 K-bytes.

### Return value

Highest address of the system stack area

### Remarks

**See also:**

## InitHeap

Initializes a heap area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.0* | *7/31/96* |

### Syntax

**void InitHeap** (**head*, *size*)
**unsigned long** **head*;
**unsigned long** *size*;

### Arguments

| | |
|---|---|
| *head* | Pointer to heap start address |
| *size* | Heap size (a multiple of 4, in bytes) |

### Explanation

This function initializes a group of standard function library memory control functions . After using this function, malloc(), etc. are usable.

There is an overhead so the entire size in bytes cannot be used.

### Return value

None.

### Remarks

To be executed in a critical section. If several executions of this function overlap, the memory control information previously held will be lost.

**See also:** malloc (see libc/libc2).

## InitHeap2*

Initializes heap area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.6* | *10/23/96* |

### Syntax

**void InitHeap2** (**head*, *size*)
**void** **head*;
**long** *size*;

### Arguments

| | |
|---|---|
| *head* | Pointer to heap start address |
| *size* | Heap size (a multiple of 4, in bytes) |

### Explanation

This function initializes a group of standard function library memory control functions. After using this function, malloc2(), etc. are usable.

There is an overhead so the entire "size" in bytes cannot be used. This is the bug fix version of InitHeap() but has larger program size since this is a memory resident function

### Return value

None.

### Remarks

If several executions of this function overlap, the memory control information previously held will be lost.

**See also:** InitHeap(),malloc2(),realloc2(),calloc2(),free2()

## ioctl

Controls devices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long ioctl** (*fd*, *com*, *arg*)
**int** *fd*;
**int** *com*;
**int** *arg*;

### Arguments

| | |
|---|---|
| *fd* | File descriptor |
| *com* | Control command |
| *arg* | Control command argument |

### Explanation

Executes all types of control commands on the device. Details of the commands and their arguments are given separately for each device.

### Return value

Returns the value “1” if it succeeds and the value “0” otherwise.

### Remarks

**See also:** open (p. 1-46).

## Krom2RawAdd

Collects Kanji font pattern addresses.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long Krom2RawAdd** (*sjiscode*)
**unsigned short** *sjiscode*;

### Arguments

| | |
|---|---|
| *sjiscode* | Shift JIS code |

### Explanation

This function acquires the starting address in the kernel of the font pattern corresponding to the Kanji character specified by *sjis code*.

### Return value

The starting address of a Kanji font pattern is returned. If there is no font data corresponding to the specified Kanji character, a value of -1 is returned.

### Remarks

**See also:**

## Krom2RawAdd2

Collects Kanji font pattern addresses.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.2* | *7/31/96* |

### Syntax

**unsigned long Krom2RawAdd2** (*sjiscode*)
**unsigned short** *sjiscode*;

### Arguments

| | |
|---|---|
| *sjiscode* | Shift JIS code |

### Explanation

Acquires the head address in the font pattern kernel corresponding to the non-Kanji/Kanji No. 1 level/foreign language specified by the *sjiscode.*

### Return value

Returns the kanji font pattern head address.

Returns -1 when the font data corresponding to the specified kanji is not prepared.

### Remarks

**See also:**

## Load

Loads an execute file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long Load** (**name*, **exec*)
**char** **name*;
**struct EXEC** **exec*;

### Arguments

| | |
|---|---|
| *name* | Pointer to filename |
| *exec* | Pointer to execute file information |

### Explanation

This function reads the PS-X EXE format file *name* to the address specified by its internal header, and writes internal information to *exec*.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

This function needs the ISO 9660 file system to run properly. _96_init() must be called to initialize this system and _96_remove must be called to exit this system. Calls FlushCache () internally.

**See also:** EXEC structure (p. 1-4), Exec (p.1-22), _96_init (p. 1-71), _96_remove (p. 1-72).

## LoadExec

Executes a file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**Void LoadExec (****name***,** *s_addr***,** *s_size***)**
**char** **name*;
**unsigned long** *s_addr*;
**unsigned long** *s_size*;

### Arguments

| | |
|---|---|
| *name* | Pointer to a PS-X EXE format execution file name (fewer than 19 characters) |
| *s_addr* | Stack area starting address |
| *s_size* | Number of bytes in stack area |

### Explanation

This function calls Load() and Exec(), then reads a file name into memory and executes the file. *s_addr* and *s_size* are passed to Exec() and set by the structure EXEC.

### Return value

None. There is no return value when the function executes normally.

### Remarks

This function needs the ISO 9660 file system to run properly. _96_init() must be called to initialize this system and _96_remove must be called to exit this system.

**See also:** EXEC (p. 1-4), Load (p. 1-39), Exec (p. 1-22), _96_init (p. 1-71), _96_remove (p. 1-72).

## LoadTest

Load test.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long LoadTest** (**name*, *exec*)
**char** **name*;
**struct EXEC** **exec*;

### Arguments

| | |
|---|---|
| *name* | Pointer to filename |
| *exec* | Pointer to data in an execute file |

### Explanation

This function writes internal information from a PS-X EXE format file *name* to *exec*.

### Return value

On success, the function returns the executione starting address. On failure, it returns 0.

### Remarks

**See also:** EXEC (p. 1-4), Load (p. 1-39).

## lseek

Moves a file pointer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long lseek** (*fd*, *offset*, *flag*)
**int** *fd*;
**unsigned int** *offset*;
**int** *flag*;

### Arguments

| | |
|---|---|
| *fd* | File descriptor |
| *offset* | Offset |
| *flag* | Start point flag |

### Explanation

This function moves a file pointer to the device indicated by the descriptor *fd*. *offset* stands for the number of bytes to be moved. The starting point of the movement varies with the value of the *flag*. However, it does not apply to a tty driver. Any of the following can be designated as *flag*:

**Table 1-3**

| flag macro | Operation |
|---|---|
| SEEK_SET | Start of file |
| SEEK_CUR | Current position |

### Return value

On success, the function returns the current file pointer. On failure, it returns -1.

### Remarks

**See also:** open (p. 1-46), read (p. 1-49), write (p. 1-70).

## malloc2*

Allocates main memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.6* | *10/23/96* |

### Syntax

**#include** *<stdlib.h>*
**void** **malloc2(size_t s)*

### Arguments

*<stdlib.h>*
**malloc2*
*(size_t s)*

### Explanation

This function allocates s bytes of memory block from the heap memory. Corresponds to InitHeap2().

### Return value

Returns a pointer to allocated memory block. If failed, NULL is returned. *Heap memory is defined as below:

| | |
|---|---|
| Low Address | Module Highest Address + 4 |
| High Address | On-board memory - 64KB |

### Remarks

**See also:** calloc2(),realloc2(),free2()

## nextfile

Looks up the next file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**struct DIRENTRY *nextfile** (**dir*)
**struct DIRENTRY** **dir*;

### Arguments

*dir* Pointer to a buffer holding information relating to the referenced file.

### Explanation

This function continues the lookup under the same conditions as the function "firstfile()", executed immediately beforehand. If it finds the corresponding file, it stores information relating to this file in *dir*.

### Return value

Returns *dir* if it succeeds, and “0” otherwise.

### Remarks

If the shell cover of the CD-ROM drive has been opened since the execution of the immediately preceding function “firstfile()”, this function fails on execution, and reports that the file has not been found.

**See also:** DIRENTRY (p. 1-3), firstfile (p. 1-24).

## open

Opens a file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long open** (**devname*, *flag*)
**char** **devname*;
**int** *flag*;

### Arguments

| | |
|---|---|
| *devname* | Pointer to a filename |
| *flag* | Open mode |

### Explanation

This function opens a device for low-level input/output, and returns the descriptor. *flag* is dependent on the device. Common parts are as follows:

**Table 1-4**

| Macro | Open mode |
|---|---|
| O_RDONLY | Read only |
| O_WRONLY | Write only |
| O_RDWR | Both read and write |
| O_CREAT | Create new file |
| O_NOBUF | Non-buffer mode |
| O_NOWAIT | Asynchronous mode |

### Return value

On success, the function returns the descriptor. On failure, it returns -1.

### Remarks

**See also:** close (p.1-12).

## OpenEvent

Opens an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long OpenEvent** (*desc*, *spec*, *mode*, **func*)
**unsigned long** *desc*;
**long** *spec*;
**long** *mode*;
**long** **func*();

### Arguments

| | |
|---|---|
| *desc* | Cause descriptor |
| *spec* | Event type |
| *mode* | Mode |
| *func* | Pointer to the handler function |

### Explanation

This function secures the EvCB for an event with the descriptor *desc* and event class *spec*.

### Return value

On success, the function returns an event descriptor. On failure, it returns -1.

### Remarks

To be executed in a critical section.

**See also:** EvCB structure (p. 1-3), CloseEvent (p. 1-14), DeliverEvent (p.1-17).

## OpenTh

Opens a thread.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long OpenTh** (**func*, *sp*, *gp*)
**unsigned long** (**func*)();
**unsigned long** *sp*;
**unsigned long** *gp*;

### Arguments

| | |
|---|---|
| *func* | Pointer to the execution start function |
| *sp* | Stack pointer value |
| *gp* | Global pointer value |

### Explanation

This function secures a TCB, and initializes it according to the arguments. This TCB can be executed using ChangeTh().

### Return value

On success, the function returns the descriptor. On failure, it returns -1.

### Remarks

To be executed in a critical section.

**See also:** TCB structure (p. 1-5), CloseTh (p. 1-11), ChangTh (p. 1-12).

## read

Reads data from a file

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**int** (*fd*, **buf*, *n*)
**int** *fd*;
**char** **buf*;
**int** *n*;

### Arguments

| | |
|---|---|
| *fd* | File descriptor |
| *buf* | Pointer to read buffer address |
| *n* | Number of bytes to be read |

### Explanation

This function reads *n* bytes from the descriptor *fd* to the area specified by *buf*.

### Return value

On normal termination, the function returns the actual number of bytes read into the area. Any other value returns -1.

### Remarks

**See also:** open (p. 1-46).

## realloc2*

Changes the heap memory allocation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.6* | *10/23/96* |

### Syntax

**#include** *<stdlib.h>*
**void** **realloc2*
(**void** **block, size_t s*)

### Arguments

*<stdlib.h>*
**block*
*size_t s*

### Explanation

This function increases/decreases the size of the memory block previously allocated to "s" bytes. Same as malloc2 when block is NULL. Corresponds to InitHeap2().

### Return value

Returns a pointer to the reallocated memory block. The new pointer may have different address from the original. If reallocation fails, NULL will be returned, and original block will not be released.

### Remarks

**See also:** calloc2(),malloc2(),free2()

## rename

Changes a file name.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long rename** (**src*, **dest*)
**char** **src*;
**char** **dest*;

### Arguments

| | |
|---|---|
| *src* | Pointer to the old filename |
| *dest* | Pointer to the new filename |

### Explanation

Changes the filename from *src* to *dest*. In both cases, the full path from the device name must be specified.

### Return value

Returns “1” if it succeeds, and “0” otherwise.

### Remarks

This function is only effective on writeable file systems.

**See also:**

## ResetCallback

Initializes all callbacks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *3.0* | *7/31/96* |

### Syntax

**void ResetCallback**()

### Arguments

None.

### Explanation

Initializes all system callbacks. Sets all callback functions to 0 (unregistered), and after securing the interrupt context stack, sets up the environment for accepting interrupts.

### Return value

None.

### Remarks

ResetCallback() must be called after program boot, before any other processing is performed.

The environment initialized by ResetCallback() will remain valid until StopCallback() is called.

It is acceptable to continuously call ResetCallback() without StopCallback(). However, the second and subsequent calls will be ignored.

**See also:**

## ResetRCnt

Resets a root counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long ResetRCnt** (*spec*)
**long** *spec*;

### Arguments

| | |
|---|---|
| *spec* | Specifies a root counter |

### Explanation

This function resets a root counter *spec* to 0.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

**See also:** SetRCnt (p. 1-58), GetRCnt (p. 1-31), StartRCnt (p. 1-61), StopRCnt (p. 1-63).

## RestartCallback

Restarts halted call-back.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *3.2* | *7/31/96* |

### Syntax

**int RestartCallback (void)**

### Arguments

None.

### Explanation

Restores the halted call-back to the status immediately prior to when it was halted.

Differs from ResetCallback () in that the call-back functions and call-back stack are not initialized.

### Return value

None.

### Remarks

ResetCallback () must be executed before executing RestartCallBack ().

The environment initialized by RestartCallback () is valid until StopCallback () is called.

There is no problem even if RestartCallback () is successively called without inserting StopCallback (), but calls from the second one onwards will be ignored.

**See also:**

## ReturnFromException

Return from exception.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void ReturnFromException** (*void*)

### Arguments

None.

### Explanation

Accesses the exception context and returns from exception processing. It is used in an event handler or callback function.

### Return value

None if the function is executed normally.

### Remarks

**See also:**

## SetConf

Modifies the kernel configuration.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long SetConf** (*ev*, *tcb*, *sp*)
**unsigned long** *ev*;
**unsigned long** *tcb*;
**unsigned long** *sp*;

### Arguments

| | |
|---|---|
| *ev* | Number of event management block (EvCB) elements |
| *tcb* | Number of task management block (TCB) elements |
| *sp* | Ignored |

### Explanation

This function modifies system configuration parameters to reconfigure the kernel configuration, specifically the allocation of the system internal table.

All the contents of event and task management blocks and all the settings for event handlers and callback functions in each library are destroyed. However, file descriptors are not affected (all the descriptors should be closed before SetConf call) because most of the device drivers are driven by the event handler.

All patches to the kernel are holded.

### Return value

1 will be returned on success of the modification. Otherwise, 0 will be returned.

### Remarks

This function should be executed at the head of the first execution file. The operations of libraries initialized before the execution of this function are not ensured.

This function eliminates the ISO-9660 file system installed in the kernel immediately after activation (call _96_init() to reinstate). The result of operations on the opened files are not predictable.

If the number of the designated elements exceeds the maximum, the operation of the system after the execution of this function is not defined.

**See also:** GetContf (p. 1-27).

## SetMem

Modifies the valid memory size.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void SetMem** (*n*)
**unsigned long** *n*;

### Arguments

| | |
|---|---|
| *n* | Valid memory size (in megabytes) |

### Explanation

This function changes the valid memory size to the value specified by the argument. *n* must be 2 (2 megabytes) or 8 (8 megabytes). Any values other than these are ignored.

### Return value

None.

### Remarks

Memory access out of the valid range results in the generation of CPU exception irrespective of the mounted physical memory.

**See also:**

## SetRCnt

Sets a root counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long SetRCnt** (*spec*, *target*, *mode*)
**long** *spec*;
**unsigned short** *target*;
**long** *mode*;

### Arguments

| | |
|---|---|
| *spec* | Root counter specification |
| *target* | Target value |
| *mode* | Mode |

### Explanation

Set the root counter in *spec*, the target value in *target*, and the mode in *mode*. If *mode* is set to RCntMdINTR, an interrupt is generated and the counter is reset once the target value is reached.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

**See also:** GetRConf (p. 1-31), StartRCnt (p. 1-61), StopRCnt (p. 1-63), ResetRCnt (p. 1-52).

## SetSp

Sets a stack pointer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long SetSp** (*new-sp*)
**unsigned long** *new-sp*;

### Arguments

| | |
|---|---|
| *new-sp* | value set in sp register |

### Explanation

Sets *new-sp* in the sp register.

### Return value

Returns the sp register value before modification.

### Remarks

**See also:** EXEC (p. 1-4), OpenTh (p. 1-48), Load (p. 1-39), Exec (p. 1-22).

## StartRCnt

Starting a root counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long StartRCnt** (*spec*)
**long** *spec*;

### Arguments

| | |
|---|---|
| *spec* | Root counter |

### Explanation

This function enables interrupts for root counter *spec*.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

**See also:** GetRCnt (p. 1-31), ResetRCnt (p. 1-52), SetRCnt (p. 1-58), StopRCnt (p. 1-63).

## StopCallback

Stops all callbacks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *3.0* | *7/31/96* |

### Syntax

**void StopCallback**()

### Arguments

None.

### Explanation

Stops all system callbacks.

### Return value

None.

### Remarks

Before terminating programs, StopCallback() must be called to disable all interrupts.

**See also:**

## StopRCnt

Stops a root counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long StopRCnt** (*spec*)
**long** *spec*;

### Arguments

| | |
|---|---|
| *spec* | Root counter |

### Explanation

This function disables interrupts for root counter *spec*.

### Return value

On success, the function returns 1. On failure, it returns 0.

### Remarks

**See also:** StartRCnt (p. 1-61), SetRCnt (p. 1-58), ResetRCnt (p. 1-52), GetRCnt (p. 1-31).

## SwEnterCriticalSection

Suppresses interrupts.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void SwEnterCriticalSection** (*void*)

### Arguments

None.

### Explanation

This function suppresses interrupts. Because no system call interrupt is generated internally, this function can be invoked in event handling and callback functions. It must be executed in a critical section.

### Return value

None.

### Remarks

**See also:** EnterCriticalSection (p. 1-23), SwExitCriticalSection (p. 1-63).

## SwExitCriticalSection

Permits interrupts.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void SwExitCriticalSection** (*void*)

### Arguments

None.

### Explanation

This function permits interrupts. Because no system call interrupt is generated internally, the function can be invoked in event handling and callback functions. It must be executed in a critical section.

### Return value

None.

### Remarks

**See also:** EnterCriticalSection (p. 1-20), SwExitCriticalSection (p. 1-63).

## SystemError

Displays the system error screen.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void SystemError** (*c*, *n*)
**char** *c*;
**long** *n*;

### Arguments

| | |
|---|---|
| *c* | Error identification character (Alphabetic character) |
| *n* | Error identification code (0 to 999) |

### Explanation

This function displays a detected system error forthe user (game player). In the PlayStation, exit() is called. Successful execution results in no return value.

### Return value

None.

### Remarks

**See also:**

## TestEvent

Testing an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long TestEvent** (*event*)
**unsigned long** *event*;

### Arguments

| | |
|---|---|
| *event* | Event descriptor |

### Explanation

This function checks to see whether or not the event specified by the descriptor event has occurred. If so, the function restores the event state to EvStACTIVE.

### Return value

If the event is found to have occurred, the function returns 1. Otherwise, it returns 0.

### Remarks

**See also:** DeliverEvent (p. 1-17), EnableEvent (p. 1-19), WaitEvent (p. 1-69), OpenEvent (p. 1-48), CloseEvent (p. 1-15), UnDeliverEvent (p. 1-69), DisableEvent (p. 1-19).

## undelete

Resurrect a file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long undelete** (**name*)
**char** **name*;

### Arguments

| | |
|---|---|
| *name* | Pointer to filename |

### Explanation

Resurrects the previously deleted file specified by *name*.

### Return value

Returns “1” if it succeeds, and “0” otherwise.

### Remarks

**See also:** delete (p. 1-16).

## UnDeliverEvent

Cancels an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void UnDeliverEvent** (*ev1*, *ev2*)
**unsigned long** *ev1*;
**long** *ev2*;

### Arguments

| | |
|---|---|
| *ev1* | Cause descriptor |
| *ev2* | Event class |

### Explanation

This function returns event state from EvStALREADY (already occurred) to EvStACTIVE if the event mode is EvMdNOINTR.

### Return value

None.

### Remarks

This function must be executed in a critical section.

**See also:** DeliverEvent (p. 1-17), EnableEvent (p. 1-19), OpenEvent (p. 1-47), TestEvent (p. 1-65), WaitEvent (p. 1-69), EnterCriticalSection (p. 1-20).

## WaitEvent

Waits for the occurrence of an event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long WaitEvent** (*event*)
**unsigned long** *event*;

### Arguments

| | |
|---|---|
| *event* | Event descriptor |

### Explanation

This function waits until an event specified by the descriptor event occurs, and returns after restoring the event state to EvStACTIVE.

### Return value

On success, the function returns 1. Otherwise, it returns 0.

### Remarks

**See also:** TestEvent (p. 1-66), OpenEvent (p. 1-48), CloseEvent (p. 1-15), DeliverEvent (p. 1-18), UnDeliverEvent (p. 1-69), DisableEvent (p. 1-19), EnableEvent (p. 1-20).

## write

Writes data to a file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**int write** (*fd*, **buf*, *n*)
**int** *fd*;
**char** **buf*;
**int** *n*;

### Arguments

| | |
|---|---|
| *fd* | File descriptor |
| *buf* | Pointer to the write buffer address |
| *n* | Number of bytes to be written |

### Explanation

This function writes *n* bytes from the descriptor *fd* to the area specified by *buf*.

### Return value

At normal termination, this function returns the number of bytes actually written to the area. Any other result returns -1.

### Remarks

**See also:** open (p. 1-46).

## _96_init

Installs the ISO-9660 file system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void _96_init** (*void*)

### Arguments

None.

### Explanation

This function installs the ISO-9660 file system driver that manages access to the CD-ROM in the kernel.

### Return value

None.

### Remarks

**See also:** _96_remove (p. 1-72).

## _96_remove

Removes the ISO-9660 file system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void _96_remove** (*void*)

### Arguments

None.

### Explanation

This function removes the ISO-9660 file system driver that manages access to the CD-ROM from the kernel.

### Return value

None.

### Remarks

**See also:** _96_init (p. 1-71).

## _boot

Reboots the system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void _boot** (*void*)

### Arguments

None.

### Explanation

This function reboots the system. This is an interface used to develop demonstration programs. Do not use it for general title applications.

### Return value

None.

### Remarks

**See also:**

## _get_errno

Collects the latest I/O error code.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**int _get_errno** (*void*)

### Arguments

None.

### Explanation

This function collects the latest error code through all file descriptors. Error codes are defined in sys/errno.h.

### Return value

Error code

### Remarks

**See also:**

## _get_error

Collects an error code for afile descriptor.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *3.0* | *7/31/96* |

### Syntax

**int_get_error** (*fd*)
**int** *fd*

### Arguments

| | |
|---|---|
| *fd* | File descriptor |

### Explanation

This function returns the code of the most recent error on the specified file descriptor. Error codes are defined in sys/errno.h.

### Return value

Error code.

### Remarks

**See also:**

# Chapter 2: “Standard” C Library
# Table of Contents

## abs

Calculates absolute value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Abs.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**int abs(***i***)**
**int** *i*;

### Arguments

| | |
|---|---|
| *i* | Integer |

### Explanation

This function calculates the absolute value of the integer *i*. This is essentially a function for finding the absolute value of an integer of the type int, but in R3000, int and long are the same size, so on this system, this function is equivalent to the function labs() described later.

### Return value

This function returns the absolute value of the argument.

### Remarks

**See also:** labs (p. 2-16).

## atoi

Converts a string to an integer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Convert.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**int atoi** (**s*)
**char** **s*;

### Arguments

| | |
|---|---|
| *s* | Pointer to a character string |

### Explanation

Converts a string to its integer equivalent. This function is the same as (long) strtol(s, (chr**) NULL). On this system, it is equivalent to atol(), described later.

### Return value

This function returns the result obtained by converting the input value *s* to an integer.

### Remarks

**See also**: atol (p. 2-4), strtol (p. 2-47).

## atol

Converts a character string to a long.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Convert.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**long atol** (**s*)
**char** **s*;

### Arguments

| | |
|---|---|
| *s* | Pointer to a character string |

### Explanation

This function is the same as(long) strtol(s, (chr**) NULL).

### Return Value

This function returns the result obtained by converting the input value *s* to a long.

### Remarks

**See also**: atoi (p. 2-3), strtol (p. 2-47).

## bcmp

Compares memory blocks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**int bcmp**(**b1*, **b2*, *n*)
**unsigned char** **b1*;
**unsigned char** **b2*;
**int** *n*;

### Arguments

| | |
|---|---|
| *b1* | Pointer to comparison source 1 |
| *b2* | Pointer to comparison source 2 |
| *n* | Number of bytes compared |

### Explanation

This function compares the first *n* bytes of *b1* and *b2*.

### Return value

The return value may be as follows, depending on the results of the comparison.

**Table 2–1**

| Result | Return value |
|---|---|
| b1<b2 | <0 |
| b1=b2 | =0 |
| b1>b2 | >0 |

### Remarks

**See also**: memcmp (p. 2-20).

## bcopy

Copies a memory block.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void bcopy(***src*, **dest*, *n*)
**unsigned char** **src*;
**unsigned char** **dest*
**int** *n*;

### Arguments

| | |
|---|---|
| *src* | Pointer to copy source |
| *dest* | Pointer to copy destination |
| *n* | Number of bytes copied |

### Explanation

This function copies the first *n* bytes of *src* to *dest*.

### Return value

None.

### Remarks

**See also**: memcpy (p. 2-21).

## bsearch

Binary search.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void *bsearch**(**key*, **base*, *n*, *w*, **fcmp*)
**unsigned char** **key*;
**unsigned char** **base*;
**size_t** *n*;
**size_t** *w*;
**int** (**fcmp*)(**unsigned char** *const_void**, **unsigned char** *const void**);

### Arguments

| | |
|---|---|
| *key* | Pointer to storage destination of the value to be searched for |
| *base* | Pointer to storage destination of the array to be searched for |
| *n* | Number of elements |
| *w* | Size of one element |
| *fcmp* | Pointer to address of comparison function |

### Explanation

This function carries out a binary search on a table of *n* items (of item size *w*) starting from base, for an item matching *key*.

### Return value

This function returns the address of the first item matching the search key. If no matching item is found, it returns 0.

### Remarks

**See also:**

## bzero

Fills a memory block with zeros.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void bzero**(**p*, *n*)
**unsigned char** **p*;
**int** *n*;

### Arguments

| | |
|---|---|
| *p* | Pointer to memory block |
| *n* | Size |

### Explanation

This function sets *n* bytes to the value 0, starting from *p*.

### Return value

None.

### Remarks

**See also:**

## calloc

Allocates main memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Malloc.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void *calloc**(*n*, *s*)
**size_t** *n*;
**size_t** *s*;

### Arguments

| | |
|---|---|
| *n* | Number of blocks |
| *s* | Size of block |

### Explanation

This function secures *n* block of *s* bytes each from the heap and clears memory allocated to 0.

### Return value

This function returns a pointer to the memory block secured. If the function fails, it returns NULL.

### Remarks

**See also**: malloc (p. 2-18), realloc (p. 2-30), free (p. 2-11).

## exit

Terminates a program normally.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdlib.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void exit(***err***)**
**int** *err*;

### Arguments

| | |
|---|---|
| *err* | Error code |

### Explanation

When this function is executed on the PlayStation itself, a system error notice window (including the error code) is displayed, and the system enters an infinite loop. When this function is executed on a development machine, the program currently being executed is terminated, and the system returns to the debug monitor.

### Return value

None.

### Remarks

**See also:**

## free

Releases allocated memory blocks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Malloc.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void free**(**block*)
**void** **block*;

### Arguments

| | |
|---|---|
| *block* | Pointer to a memory block allocated by a function such as malloc(). |

### Explanation

This function releases memory blocks secured by the functions calloc(), malloc() and realloc().

### Return value

None.

### Remarks

**See also**: calloc (p. 2-9), malloc (p. 2-18), realloc (p. 2-30).

## getc

Gets one character from the stream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdio.h* | *2.x* | *7/31/96* |

### Syntax

**#include <romio.h>**
**char getc**(**stream*)
**int** **stream*;

### Arguments

| | |
|---|---|
| *stream* | Pointer to input stream |

### Explanation

This function gets one character from the input stream.

### Return value

If this function succeeds, it returns the character it has read.

When getc reaches the end of the file, or when an error is generated, it returns EOF.

### Remarks

Devices and systems with a block size of 1 may all be used as the standard input/output stream as follows.

- Close (0);
- Close (1);
- Open (<device name>, O_RDONLY);
- Open (<device name>, O_WRONLY);

**See also**: getchar (p. 2-13), gets (p. 2-14).

## getchar

Gets one character from the standard input stream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdio.h* | *2.x* | *7/31/96* |

### Syntax

**#include <romio.h>**
**char getchar**(*void*)

### Arguments

None.

### Explanation

This function gets one character from the standard input stream. It is the same as getc(stdin).

### Return value

The return value is the same as for getc().

### Remarks

**See also**: getc (p. 2-12), gets (p. 2-14).

## gets

Reads a character string from the standard input.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdio.h* | *2.x* | *7/31/96* |

### Syntax

**#include <romio.h>**
**char** ***gets**(**s*)
**char** **s*;

### Arguments

| | |
|---|---|
| *s* | Pointer to storage destination for input character string |

### Explanation

This function reads a character string from the standard input stream (stdin) and stores it in *s* until a new-line character is read.

### Return value

If this function succeeds, it returns *s* the new-line character is discarded and a null character is written immediately after the last character read. If it reaches the end of the file, or if an error is generated, it returns NULL.

### Remarks

**See also**: getc (p. 2-12), getchar (p. 2-13).

## isXXXX...

Tests characters.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Ctype.h* | *2.x* | *7/31/96* |

### Syntax

**#include <ctype.h>**
**long isXXXX***(c)*
**long** *c*;

### Arguments

| | |
|---|---|
| *c* | Character |

### Explanation

This function tests on the character *c*. All of the tests are macros. The test conditions are as follows.

**Table 2–2**

| Name | Conditions |
|---|---|
| isalnum(c) | isapha(c) \|\| isdigit(c) |
| iasalpha(c) | isupper(c) \|\| islower(c) |
| isascii(c) | ASCII character |
| iscntrl(c) | Control character |
| isdigit(c) | Decimal |
| isgraph(c) | Printing characters other than space |
| islower(c) | Lower-case character |
| isprint(c) | Printing characters including space |
| ispunct(c) | Printing characters other than space and alphanumerics |
| ispacet(c) | Space, new page, new line, restore, tab |
| isupper(c) | Upper-case character |
| isxdigit(c) | Hexadecimal |

### Return value

This function returns a value other than 0 if the character *c* satisfies the test conditions, and returns the value 0 if it does not satisfy the test conditions.

### Remarks

**See also:**

## labs

Absolute value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Convert.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**long labs(***i***)**
**long** *i*;

### Arguments

| | |
|---|---|
| *i* | Long value |

### Explanation

This function calculates the absolute value of *i.*

### Return value

This function returns the absolute value of the argument.

### Remarks

**See also**: abs (p. 2-3).

## longjmp

Non-local jump.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Setjmp.h* | *2.x* | *7/31/96* |

### Syntax

**#include <setjmp.h>**
**void longjmp**(*p*, *val*)
**jmp_buf** *p*;
**int** *val*;

### Arguments

| | |
|---|---|
| *p* | Environment storage variable |
| *val* | setjmp() Return value |

### Explanation

This function makes a non-local jump to the destination specified by *p*.

### Return value

None. If the function executes normally, it does not return.

### Remarks

**See also**: setjmp (p. 2-31).

## malloc

Allocates main memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Malloc.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void *malloc**(*s*)
**size_t** *s*;

### Arguments

| | |
|---|---|
| *s* | Number of bytes to be allocated |

### Explanation

This function secures a block of *s* bytes from the memory heap.

### Return value

This function returns a pointer to the secured memory block. If it has failed to secure a block, it returns NULL.

Note that the memory heap is defined as follows:

Bottom address: top address of module + 4.

Top address: available memory -4.

### Remarks

**See also**: calloc (p. 2-9), realloc (p. 2-30), free (p. 2-11).

## memchr

Searches memory block for a character.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <memory.h>**
**void *memchr**(**s*, *c*, *n*)
**unsigned char** **s*;
**unsigned char** *c*;
**int** *n*;

### Arguments

|  |  |
|---|---|
| *s* | Pointer to memory block |
| *c* | Character |
| *n* | Number of bytes |

### Explanation

This function searches the memory block of *n* bytes starting from *s*, looking for the first appearance of the character c.

### Return value

This function returns a pointer to the location at which *c* was found. If *c* was not found, it returns NULL.

### Remarks

**See also:**

## memcmp

Compares memory blocks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <memory.h>**
**void *memcmp**(**s1*, **s2*, *n*)
**unsigned char** **s1*;
**unsigned char** **s2*;
**int** *n*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to comparison source memory block1 |
| *s2* | Pointer to comparison source memory block 2 |
| *n* | Number of bytes compared |

### Explanation

This function compares the first *n* bytes of *s1* and *s2*.

### Return value

This function returns the values shown below, depending on the results of the comparison of *s1* and *s2*.

**Table 2–3**

| Result | Return value |
|---|---|
| s1<s2 | <0 |
| s1=s2 | =0 |
| s1>s2 | >0 |

### Remarks

**See also**: bcmp (p. 2-5).

## memcpy

Copies memory blocks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <memory.h>**
**void *memcpy**(**dest*, **src*, *n*)
**unsigned char** **dest*;
**unsigned char** **src*;
**int** *n*;

### Arguments

| | |
|---|---|
| *dest* | Pointer to copy destination memory block |
| *src* | Pointer to copy source memory block |
| *n* | Number of bytes copied |

### Explanation

This function copies the first n bytes of *src* to *dest*.

### Return value

This function returns *dest*.

### Remarks

**See also**: bcopy (p. 2-6).

## memmove

Copies a memory block.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <memory.h>**
**void *memmove**(******dest*, **src*, *n*)
**unsigned char** **dest*;
**unsigned char** **src*;
**int** *n*;

### Arguments

| | |
|---|---|
| *dest* | Pointer to copy destination memory block |
| *src* | Pointer to copy source memory block |
| *n* | Number of bytes copied |

### Explanation

This function copies the first *n* bytes of *src* to *dest*. The block is copied correctly, even between overlapping objects.

### Return value

This function returns *dest*.

### Remarks

**See also:**

## memset

Writes specified characters to a memory block.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Memory.h* | *2.x* | *7/31/96* |

### Syntax

**#include <memory.h>**
**void *memset**(**s*, *c*, *n*)
**unsigned char** **s*;
**unsigned char** *c*;
**int** *n*;

### Arguments

| | |
|---|---|
| *s* | Pointer to memory block |
| *c* | Character |
| *n* | Number of characters |

### Explanation

This function writes *c* to a memory block of *n* bytes starting at *s*.

### Return value

This function returns *s*.

### Remarks

**See also:**

## printf

Formatted output.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdio.h* | *2.x* | *7/31/96* |

### Syntax

**#include <romio.h>**
**long printf**(*const char *fmt*[, *argument ...*])

### Arguments

|  |  |
|---|---|
| *fmt* | Pointer to input format character string |
| *argument* | Argument corresponding to *fmt* |

### Explanation

Omitted. See a C language reference. Conversion directives f, e, E, g and G cannot be used.

### Return value

printf returns the length of the output character string. If an error is generated, the function returns NULL.

### Remarks

**See also:**

## putc

Outputs one character to the stream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdio.h* | *2.x* | *7/31/96* |

### Syntax

**#include <romio.h>**
**void putc**(*c*, *stream*)
**char** *c*;
**int** *stream*;

### Arguments

| | |
|---|---|
| *c* | Output character |
| *stream* | Output stream |

### Explanation

This function outputs a character *c* to the output stream.

### Return value

This function returns *c* if it succeeds, and EOF if an error is generated.

### Remarks

**See also**: putchar (p. 2-26), puts (p. 2-27).

## putchar

Outputs one character to the standard output stream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdio.h* | *2.x* | *7/31/96* |

### Syntax

**#include <romio.h>**
**void putchar**(*c*)
**char** *c*;

### Arguments

| | |
|---|---|
| *c* | Output character |

### Explanation

This function outputs a character *c* to the standard output. It is the same as putc(stdout).

### Return value

The return value is the same as for putc().

### Remarks

**See also**: putc (p. 2-25), puts (p. 2-27).

## puts

Outputs a character string to the standard output stream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Stdio.h* | *2.x* | *7/31/96* |

### Syntax

**#include <romio.h>**
**void puts**(**s*)
**char char** **s*

### Arguments

| | |
|---|---|
| *s* | Pointer to output character string |

### Explanation

This function outputs a character string ending in NULL to the standard output stream (stdout), and finally outputs a newline character.

### Return value

This function returns a non-negative value if it succeeds, and EOF if an error is generated.

### Remarks

**See also**: putc (p. 2-25), putchar (p. 2-26).

## qsort

Quick sort.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Rand.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**void qsort** (**base*, *n*, *w*, **fcmp*)
**void** **base*;
**size_t** *n*;
**size_t** *w*;
**int** (**fcmp*)(*const void**, *const void **)

### Arguments

| | |
|---|---|
| *base* | Pointer to storage destination of array to be sorted |
| *n* | Number of elements |
| *w* | Size of on element |
| *fcmp* | Pointer to address of comparison function |

### Explanation

This function quick-sorts a table of *n* items (of item size *w*) starting with base, with *fcmp* as the comparison function.

### Return value

None.

### Remarks

**See also:**

## rand

Generates random numbers.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Rand.h* | *2.x* | *7/31/96* |

### Syntax

**#include <rand.h>**
**int rand**(*void*)

### Arguments

None.

### Explanation

This function generates a pseudo-random number from 0 to RAND_MAX (0x7FFF=32767).

### Return value

This function returns the pseudo-random number which has been generated.

### Remarks

**See also**: srand (p. 2-32).

## realloc

Changing heap memory allocations.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Malloc.h* | *2.x* | *7/31/96* |

### Syntax

**#include <memory.h>**
**void** ***realloc**(**block*, *s*)
**void** **block*;
**size_t** *s*;

### Arguments

| | |
|---|---|
| *block* | Pointer to a block secured by a function such as malloc() |
| *s* | New size |

### Explanation

This function takes a previously concerned *block* and contracts it or expands it to *s* bytes. If block is NULL, this function works in the same way as malloc.

### Return value

This function returns the address of the reallocated block. This address may be different to the old address.

If it fails to perform the allocation, the function returns NULL. In this case, the old block is not released.

### Remarks

**See also**: calloc (p. 2-9), malloc (p. 2-18), free (p. 2-11).

## setjmp

Defines non-local jump destination.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Setjmp.h* | *2.x* | *7/31/96* |

### Syntax

**#include <setjmp.h>**
**long setjmp**(*p*)
**jmp_buf** *p*;

### Arguments

| | |
|---|---|
| *p* | Environment storage variable |

### Explanation

This function stores the destination information for a non-local jump at p. If longjmp(p, val) is executed, the system will return from setjmp().

### Return value

This function returns the value given to the second argument of longjmp() when the jump is executed.

### Remarks

**See also**: longjmp (p. 2-17).

## srand

Initializes the random number generator.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Rand.h* | *2.x* | *7/31/96* |

### Syntax

**#include <rand.h>**
**void srand**(*seed*)
**unsigned long** *seed*;

### Arguments

| | |
|---|---|
| *seed* | Random number seed |

### Explanation

This function sets a new starting point for random number generation. The default is 1.

### Return value

None.

### Remarks

**See also**: rand (p. 2-29).

## strcat

Concatenates character strings.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strcat**(**dest*, **src*)
**char** **dest*;
**char** **src*;

### Arguments

| | |
|---|---|
| *dest* | Pointer to concatenation target string |
| *src* | Pointer to concatenation source string |

### Explanation

This function appends the character string *src* to the end of the character string *dest*.

### Return value

This function returns *dest*.

### Remarks

**See also**: strncat (p. 2-39).

## strchr

Searches for the first location at which a specified character appears in a character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strchr**(**s*, *c*)
**char** **s*
**char** *c*;

### Arguments

| | |
|---|---|
| *s* | Pointer to character string searched |
| *c* | Character searched for |

### Explanation

This function searches for the first location at which the character *c* appears in the character string *s*.

### Return value

This function returns the address of the location at which *c* appears. If *c* has not been found, it returns NULL.

### Remarks

**See also**:

## strcmp

Compares character strings.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**int strcmp**(**s1*, **s2*)
**char** **s1*;
**char** **s2*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to character string 1 |
| *s2* | Pointer to character string 2 |

### Explanation

This function compares the character string *s2* with the character string *s1*, treating each character as an unsigned char.

### Return value

This function returns one of the values shown below, depending on the comparison result.

**Table 2–4**

| Result | Return value |
|---|---|
| s1<s2 | <0 |
| s1=s2 | =0 |
| s1>s2 | >0 |

### Remarks

**See also**:

## strcpy

Copies a character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strcpy**(**dest*, **src*)
**char** **dest*;
**char** **src*;

### Arguments

| | |
|---|---|
| *dest* | Pointer to copy destination character string |
| *src* | Pointer to copy source character string |

### Explanation

This function copies the character string *src* to the character string *dest*.

### Return value

This function returns *dest*.

### Remarks

**See also**: strncpy (p. 2-41).

## strcspn

Search for a partial character string made up solely of characters not included in the specified character set.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**int strcspn**(**s1*, **s2*)
**char** **s1*;
**char** **s2*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to character string |
| *s2* | Pointer to character group |

### Explanation

This function returns the length of the first part of the character string *s1* consisting only of characters not included in the character string *s2*.

### Return value

This function returns the length of the partial character string found.

### Remarks

**See also**:

## strlen

Counts the number of characters in a character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**int strlen**(**s*)
**char** **s*;

### Arguments

| | |
|---|---|
| *s* | Pointer to character string |

### Explanation

This function counts the number of characters in a character string *s*.

### Return value

This function returns the number of characters.

### Remarks

**See also**:

## strncat

Concatenates character strings.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strncat**(**dest*, **src*, *n*)
**char** **dest*;
**char** **src*;
**int** *n*;

### Arguments

| | |
|---|---|
| *dest* | Pointer to concatenation destination array |
| *src* | Pointer to concatenation source character string |
| *n* | Number of characters concatenated |

### Explanation

This function appends the first *n* characters from *src* to the end of the character string dest.

### Return value

This function returns *dest*.

### Remarks

**See also**:

## strncmp

Compares character strings.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**int strcmp**(**s1*, **s2*, *n*)
**char** **s1*;
**char** **s2*;
**int** *n*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to character string 1 |
| *s2* | Pointer to character string 2 |
| *n* | Number of characters compared |

### Explanation

This function compares the first *n* characters of *s1* and *s2*, treating each character as unsigned char.

### Return value

This function returns one of the following values, depending on the comparison result (the values are the same as for strcmp).

**Table 2–5**

| Result | Return value |
|---|---|
| s1<s2 | <0 |
| s1=s2 | =0 |
| s1>s2 | >0 |

### Remarks

**See also**:

## strncpy

Copies a character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strncpy**(**dest*, **src*, *n*)
**char** **dest*;
**char** **src*;
**int** *n*;

### Arguments

| | |
|---|---|
| *dest* | Pointer to copy destination array |
| *src* | Pointer to copy source character string |
| *n* | Number of bytes |

### Explanation

This function copies *n* bytes worth of *src* to the character string *dest*. When the number of characters copied reaches *n*, the copying is terminated.

### Return value

This function returns *dest*.

### Remarks

**See also**:

## strpbrk

Searches for the first occurrence of a character within a specified character set.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strpbrk**(**s1*, **s2*)
**char** **s1*;
**char** **s2*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to character string searched |
| *s2* | Pointer to character group |

### Explanation

This function searches for the first location at which any of the characters contained in the character string *s2* appear within the character string *s1*.

### Return value

This function returns the address of the character found. If no character was found, it returns NULL.

### Remarks

**See also**:

## strrchr

Searches for the last location of a specified character in a character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strrchr**(**s*, *c*)
**char** **s*;
**char** *c*;

### Arguments

| | |
|---|---|
| *s* | Pointer to character string searched |
| *c* | Character searched for |

### Explanation

This function searches for the last occurrence of the character *c* within the character string *s*.

### Return value

This function returns the address of *c*. If *c* does not occur, it returns NULL.

### Remarks

**See also**:

## strspn

Searches for the part of a character string consisting solely of characters contained in the specified character set.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**int strspn**(**s1*, **s2*)
**char** **s1*;
**char** **s2*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to character string |
| *s2* | Pointer to character group |

### Explanation

This function returns the length of the first part of the character string *s1* which consists solely of characters included in the character string *s2*.

### Return value

This function returns the length of the partial character string it has found.

### Remarks

**See also**:

## strstr

Searches for the location of a specified partial character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strstr**(**s1*, **s2*)
**char** **s1*;
**char** **s2*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to character string searched |
| *s2* | Pointer to character string searched for |

### Explanation

This function searches for the first location of character string *s2* within character string *s1*.

### Return value

This function returns the address of *s2.* If it was not found, the function returns NULL.

### Remarks

**See also**:

## strtok

Searches for a character string demarcated by certain characters within a specified character set.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Strings.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**char** ***strtok**(**s1*, **s2*)
**char** **s1*;
**char** **s2*;

### Arguments

| | |
|---|---|
| *s1* | Pointer to character string searched |
| *s2* | Pointer to separator characters |

### Explanation

This function treats character string *s1* as a set of tokens punctuated by one or more characters from the separator character string *s2*. The first call in the sequence searches *s1* for the first character that is not contained within *s2*.

The first time strtok() is called, the starting address of the first token of s1 is returned, and a NULL character is written in immediately after this token. The address of *s1* is stored in the function, and then, when strtok() is called with NULL entered as the first argument, a search is carried out until there are no tokens left in the character string *s1*.

### Return value

This function returns the starting address of the tokens found in *s1*. If it does not find any s1 tokens, strtok() returns NULL.

### Remarks

**See also**:

## strtol

Performs long conversion of a character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Convert.h* | *2.x* | *7/31/96* |

### Syntax

**#include <strings.h>**
**long strtol**(**s*, ***endp*)
**char** **s*;
**char** ***endp*;
**unsigned int** *base*;

### Arguments

| | |
|---|---|
| *s* | Pointer to character string |
| *endp* | Storage destination of pointer to a non-convertible character string |
| *base* | Radix specification |

### Explanation

This function converts a character string *s* to long type (the same as int type in R3000). *s* must be formatted as follows.

[ws][sn][ddd]

- [ws] white space (may be omitted)
- [sn] sign (may be omitted)
- [ddd] number string (may be omitted)

The value of base determines the format of [ddd]. The letters a (or A) thru z (or Z) are ascribed values from 10-35. Only values less than base may be included in [ddd]. For some values of base, optional characters may precede the sequence of letters and digits following the sign (if present).

**Table 2–6**

| Base value | Optimal characters |
|---|---|
| 2 | 0b, 0B |
| 8 | “O,” “o” |
| 16 | 0x, 0X |

The function strtol() stops converting when it encounters a non-convertible character, and if *endp* is not NULL, it sets *endp* as the pointer to the character at which it stopped converting.

### Return value

This function returns the result obtained by converting the input value *s* to a long. If an error is generated, it returns 0.

### Remarks

**See also**: strtoul (p. 2-47).

## strtoul

Performs unsigned long conversion of a character string.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Convert.h* | *2.x* | *7/31/96* |

### Syntax

**#include <stdlib.h>**
**unsigned long strtoul**(**s*, *endp*, *base*)
**char** **s*;
**char** ***endp*;
**int** *base*;

### Arguments

| | |
|---|---|
| *s* | Pointer to character string |
| *endp* | Storage destination of pointer to a non-convertible character string |
| *base* | Radix specification |

### Explanation

This function converts a character string *s* to unsigned long type (the same as unsigned int type in R3000). s must be formatted as follows.

[ws][sn][ddd]

- [ws] white space (may be omitted)
- [sn] sign (may be omitted)
- [ddd] number string (may be omitted)

The value of base determines the format of [ddd]. The letters a (or A) thru z (or Z) are ascribed values from 10-35. Only values less than base may be included in [ddd]. For some values of base, optional characters may precede the sequence of letters and digits following the sign (if present).

**Table 2–7**

| Base value | Optimal characters |
|---|---|
| 2 | 0b, 0B |
| 8 | “O,” “o” |
| 16 | 0x, 0X |

The function strtoul() stops converting when it encounters a non-convertible character, and if *endp* is not NULL, it sets *endp* as the pointer to the character at which it stopped converting.

### Return value

This function returns the result obtained by converting the input value *s* to a long.

### Remarks

**See also**: strtol (p. 2-47).

## toascii

Masks bit 7 of the input value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Ctype.h* | *2.x* | *7/31/96* |

### Syntax

**#include <ctype.h>**
**long toascii**(*c*)
**long** *c*;

### Arguments

| | |
|---|---|
| *c* | Value |

### Explanation

This is a macro which masks the 7th bit.

### Return value

This macro returns a value obtained by masking the 7th bit of the input value *c*.

### Remarks

**See also**:

## tolower

Converts a letter to lower-case.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Ctype.h* | *2.x* | *7/31/96* |

### Syntax

**#include <ctype.h>**
**long tolower**(*c*)
**long** *c*;

### Arguments

| | |
|---|---|
| *c* | Character |

### Explanation

This macro converts a character *c* to lower case. The behavior of this macro when it is given a value not an upper-case letter is undefined.

### Return value

This macro returns a lower-case letter that corresponds to *c*.

### Remarks

**See also**:

## toupper

Converts a character to upper case.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libc\Libc2.lib* | *Ctype.h* | *2.x* | *7/31/96* |

### Syntax

**#include <ctype.h>**
**long toupper**(*c*)
**long** *c*;

### Arguments

| | |
|---|---|
| *c* | Character |

### Explanation

This macro converts a character *c* to upper case. The behavior of this macro when it is given a value not a lower-case letter is undefined.

### Return value

This macro returns an upper-case letter that corresponds to the character *c*.

### Remarks

**See also**:

# Chapter 3: Math Library
## Table of Contents

## acos

Arccosine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double acos**(x)
**double** *x*;

### Argument

*x* Value whose arccosine is to be determined, ranging from -1 to 1

### Explanation

Determines the arccosine function of *x*.

### Return value

Arccosine function of *x*, ranging from 0 to pi. Error processing is shown as follows:

**Table 3–1**

| Conditions | Return value | Error |
|---|---|---|
| fabs(x)>1 | 0 | Domain error |

### Remarks

**See also:** cos (p. 3-8), asin() (p. 3-3), atan() (p. 3-4).

## asin

Arcsine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double asin**(x)
**double** *x*;

### Argument

*x* Value whose arcsine is to be determined, ranging from -1 to 1.

### Explanation

Determines the arcsine function of *x*.

### Return value

Arcsine function of *x*, ranging from -pi/2 to pi/2.

Error processing is as follows:

**Table 3–2**

| Conditions | Return value | Error |
|---|---|---|
| fabs(x)>1 | 0 | Domain error |

### Remarks

**See also:** sin (p. 3-22), acos (p. 3-3), atan (p. 3-4).

## atan

Arctangent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double atan**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Value whose arctangent is to be calculated |

### Explanation

Determines the arctangent function of *x*.

### Return value

Arctangent function of *x*, ranging from -pi/2 to pi/2 radians.

### Remarks

**See also:** tan (p. 3-26), asin (p. 3-3), acos (p. 3-3), atan2 (p. 3-5).

## atan2

Arctangent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double atan2**(x, *y*)
**double** *x*, *y*;

### Arguments

*x, y* Floating-point value

### Explanation

Determines the arctangent of *x/y*.

### Return value

Arctangent function of *x/y*, ranging from -pi to pi.

### Remarks

If *x* and *y* are 0, a value of 0 is returned.

**See also:** atan() (p. 3-4)

## atof

Converts a string to a floating-point equivalent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double atof** (**char** **s*)

### Arguments

| | |
|---|---|
| *s* | Pointer to a string |

### Explanation

Converts a string "s" to its floating-point (double type) equivalent.

### Return value

Returns the result from converting input string "s" to a floating point equivalent in double type. When the converted value overflows, either +HUGE_VAL(1.797693134862316e+308) or -HUGE_VAL depending on the sign, will be returned. 0 is returned for underflow case.

### Remarks

Error handling is as follows:

**Table 3–3**

| Condition | Returned Value | Error Type |
|---|---|---|
| Overflow | +/- HUGE_VAL | Region Error |
| Underflow | 0 | Region Error |

**See also**: strtod (p. 3-27).

## ceil

Minimum integer not less than x (ceiling function).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double ceil**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Floating-point value |

### Explanation

This function determines the minimum integer (double type) not less than *x*.

### Return value

Minimum integer (double type) not less than *x*

### Remarks

**See also:** floor (p. 3-12).

## cos

Cosine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double cos**(x)
**double** *x*;

### Argument

*x* Angle in radians

### Explanation

Determines the cosine function of *x*.

### Return value

Cosine function of *x* (cos(x))

### Remarks

**See also:** sin (p. 3-22), tan (p. 3-26), acos (p. 3-3).

## cosh

Hyperbolic cosine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double cosh**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Angle in radians |

### Explanation

Determines the hyperbolic cosine function of *x*.

### Return value

Hyperbolic cosine function of *x* (cosh(x))

### Remarks

**See also:** sinh (p. 3-23), tanh (p. 3-28).

## exp

Exponent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double exp**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Floating-point value |

### Explanation

This function determines the exponential function of *x*.

### Return value

e raised to the x-th power (e**x)

### Remarks

**See also:** pow (p. 3-20), log (p. 3-17).

## fabs

Absolute value (macro).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double fabs**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Floating-point value |

### Explanation

This function determines an absolute value.

### Return value

Absolute value of *x.*

### Remarks

This is a macro.

**See also**:

## floor

Maximum integer not more than x (lower function).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double floor**(x)
**double** *x*;

### Argument

*x* Floating-point value

### Explanation

This function determines the maximum integer (double type) not more than *x*.

### Return value

Maximum integer not more than *x* (double type)

### Remarks

**See also:** ceil (p. 3-5).

## fmod

Floating-point remainder resulting from x/y.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double fmod**(x, *y*);
**double** *x*, *y*;

### Arguments

| | |
|---|---|
| *x* | Floating-point value |
| *y* | Floating-point value |

### Explanation

This function determines the floating-point remainder resulting from *x/y*.

### Return value

Floating-point remainder resulting from *x/y*.

If *y* is 0, a value of 0 is returned.

### Remarks

The sign of the return value is the same as of *x*.

**See also**:

## frexp

Resolution into normalized decimal part and the part raised to the second power.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double frexp**(x, **n*)
**double** *x*;
**int** **n*;

### Arguments

| | |
|---|---|
| *x* | Floating-point value |
| *n* | Pointer to a buffer for storing the part raised to the second power |

### Explanation

This function resolves *x* into a decimal portion normalized at [1/2, 1) and the portion that is raised to the second power. The decimal part is returned, and the part raised to the second power is stored in *n*.

### Return value

Normalized decimal part. [1/2, 1).

### Remarks

A pair of square brackets [] indicates a closed area, while a pair of parentheses () indicates an open area.

**See also**:

## hypot

Absolute value of a complex number.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double hypot**(*x*, *y*)
**double** *x*, *y*;

### Arguments

*x, y* Floating-point value

### Explanation

This function computes the square root of the sum of the squares of *x* and *y*.

### Return value

Square root of the sum of (x**2) and (y**2).

### Remarks

**See also**:

## ldexp

Calculates a real number from a mantissa and an exponent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double ldexp**(x, *n*)
**double** *x*;
**long** *n*;

### Arguments

| | |
|---|---|
| *x* | Floating-point value |
| *n* | Integral exponent |

### Explanation

This function determines a real number from a mantissa and an exponent.

### Return value

Value of *x* multiplied by 2 raised to the $n^{th}$ power.

### Remarks

**See also**:

## log

Natural logarithm.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double log**(x)
**double** *x*;

### Argument

*x* Value subjected to logarithmic operation

### Explanation

Determines the natural logarithmic function of *x*.

### Return value

Logarithm of x (ln(x)).

*x* must be greater than zero. Otherwise, a domain error results. Error processing is as follows.

**Table 3–4**

| Conditions | Return value | Error |
|---|---|---|
| x<0 | 0 | Domain error |
| x==0 | 1 | Range error |

### Remarks

**See also:** exp (p. 3-10), log10 (p. 3-18).

## log10

Logarithm whose base is 10.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double log10**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Value subjected to logarithmic operation |

### Explanation

This function determines the logarithmic function of x whose base is 10.

### Return value

Logarithm of x whose base is 10 (log10(x))

*x* must be greater than zero. Otherwise, an error results. Error processing is as follows.

**Table 3–5**

| Conditions | Return value | Error |
|---|---|---|
| x<0 | 0 | Domain error |
| x==0 | 1 | Range error |

### Remarks

**See also:** log (p. 3-17).

## modf

Separation into integral and fractional parts.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double modf**(x, **y*)
**double** *x*, **y*;

### Arguments

| | |
|---|---|
| *x* | Floating-point value |
| *y* | Pointer to a buffer for storing the integral part |

### Explanation

This function separates x into integral and fractional parts. The integral part is stored in *y*, with the return value being the fractional part.

### Return value

Decimal part of *x*

### Remarks

The signs of both the integral and decimal parts are the same as *x*.

**See also**:

## pow

x raised to the y-th power.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double pow**(x, *y*)
**double** *x*;
**double** *y*;

### Arguments

| | |
|---|---|
| *x* | Numerical value |
| *y* | Power |

### Explanation

This function raises *x* to the *y*-th power.

### Return value

*x* raised to the *y*-th power(x**y).

Error processing is as follows.

**Table 3–6**

| Condition | Return value | Error |
|---|---|---|
| x==0 && y>0 | 0 | |
| x==0 && y<=0 | 1 | Domain error |
| x<0 && [y not an integer] | 0 | Domain error |

### Remarks

**See also:** exp (p. 3-10).

## printf2

Formats output to console.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**long printf** (**const char** **fmt*, [*argument...*])

### Arguments

| | |
|---|---|
| *fmt* | Pointer to input format character string |
| *argument* | Argument for *fmt* |

### Explanation

Refer to a standard C language reference.

### Return value

Output character length is returned.

### Remarks

Conversion directives f, e, E, g and G may be used.

**See also**:

## sin

Sine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double sin**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Angle in radians |

### Explanation

Determines the sine function of *x*.

### Return value

Sine function of x (sin(x))

### Remarks

**See also:** cos (p. 3-8), tan (p. 3-26), asin (p. 3-3).

## sinh

Hyperbolic sine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double sinh**(x)
**double** *x*;

### Argument

*x* Angle in radians

### Explanation

Determines the hyperbolic sine function of *x*.

### Return value

Hyperbolic sine function of *x* (sinh(x))

### Remarks

**See also:** cosh (p. 3-9), tanh (p. 3-28).

## sprintf2

Format output to anarray (corresponding to floating-point and double-precision arguments).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Stdio.h* | *3.0* | *7/31/96* |

### Syntax

**long sprintf**(**char** **s*, **const char** **fmt*, [*argument...*])

### Arguments

| | |
|---|---|
| *s* | Pointer to storage destination of converted character string |
| *fmt* | Pointer to input format character string |
| *argument* | Argument for *fmt* |

### Explanation

Refer to a standard C language reference.

### Return value

Output character length is returned.

### Remarks

Conversion directives f, e, E, g and G may be used.

**See also**:

## sqrt

Square root.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double sqrt**(x)
**double** *x*

### Argument

| | |
|---|---|
| *x* | Non-negative floating-point value |

### Explanation

This function determines the non-negative square root of *x*.

### Return value

Square root of *x*.

### Remarks

Error processing is as follows.

**Table 3–7**

| Condition | Return value | Error |
|---|---|---|
| x<0 | 0 | Domain error |

**See also**:

## strtod

Converts a string to a floating-point equivalent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double strtod** (**char** **s*, **char** ***endp*)

### Arguments

| | |
|---|---|
| *s* | Pointer to a string |
| *endp* | Holds a pointer to a string that was unable to be converted |

### Explanation

This function converts a string to a double type floating-point equivalent.

"s" must be one of the following:

[ws][sn][ddd]

- [ws] White space (may be omitted)
- [sn] Sign (may be omitted)
- [ddd] Number string (may be omitted)

Stops converting upon encountering a character that was unable to be converted. If endp is not NULL, the pointer to the character in error is set to endp.

### Return value

Returns the result from converting input string "s" to a floating point in double type. When the converted value overflows, either +HUGE_VAL(1.797693134862316e+308) or -HUGE_VAL according to the sign, will be returned. 0 is returned for underflow case. If no conversion could be performed, 0 is returned.

### Remarks

Error handling is as follows:

**Table 3–8**

| Condition | Returned Value | Error Type |
|---|---|---|
| Overflow | +/- HUGE_VAL | Area Error |
| Underflow | 0 | Area Error |

**See also:** atof (p. 3-7).

## tan

Tangent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double tan**(x)
**double** *x*;

### Argument

*x* Angle in radians

### Explanation

Determines the tangent function of x.

### Return value

Tangent function of x (tan(x))

### Remarks

**See also:** sin (p. 3-22), cos (p. 3-8), atan (p. 3-4).

## tanh

Hyperbolic tangent.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libmath.lib* | *Libmath.h* | *3.0* | *7/31/96* |

### Syntax

**double tanh**(x)
**double** *x*;

### Argument

| | |
|---|---|
| *x* | Angle in radians |

### Explanation

Determines the hyperbolic tangent function of *x*.

### Return value

Hyperbolic tangent function of *x* (tanh(x))

### Remarks

**See also:** sinh (p. 3-23), cosh (p. 3-9).

# Chapter 4: Memory Card Library
## Table of Contents

## InitCARD

Initializes memory card BIOS

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**void InitCARD**(*val*)
**long** *val*;

### Arguments

| | |
|---|---|
| *val* | Indicates sharing with controller<br>0: Not shared<br>1: Shared |

### Explanation

Initializes the memory card BIOS and enters an idle state. Specify in *val* whether or not there is sharing with the controller.

When the BIOS is subsequently put into operation by StartCARD(), the low-level interface function that starts _card can be used directly.

The memory card file system uses these interfaces internally, so InitCARD needs to be executed before _bu_init().

There is no effect on the controller.

### Return value

None.

### Remarks

**See also:** bu_init (p. 4-5).

## StartCARD

Starts memory card BIOS.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**void StartCARD**(*void*)

### Arguments

None.

### Explanation

Changes the memory card BIOS initialized by InitCARD() to a run state.

Performs ChangeClearPAD(1) internally.

### Return value

None.

### Remarks

**See also:** InitCARD (p. 4-3), StopCARD (p. 4-4), _bu_init (p. 4-5), ChangeClearPAD (see libapi).

## StopCARD

Stops memory card BIOS.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**void StopCARD**(*void*)

### Arguments

None.

### Explanation

Changes memory card BIOS to an idle state (the same state as that immediately after executing InitCARD())

Performs ChangeClearPAD(1) internally.

### Return value

None.

### Remarks

**See also:** InitCARD (p. 4-3), StartCARD (p. 4-3), _bu_init (p. 4-5), ChangeClearPAD (see libapi).

## _bu_init

Initializes memory card file system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**void_bu_init**(*void*)

### Arguments

None.

### Explanation

Initializes the memory card file system.

The initialization routine does not execute automatically, so this function is required to explicitly initialize the file system.

### Return value

None.

### Remarks

**See also:** InitCARD (p. 4-3), StartCARD (p. 4-3), StopCARD (p. 4-4).

## _card_auto

Sets automatic format function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long _card_auto**(*val*)
**long** *val*;

### Arguments

| | |
|---|---|
| *val* | Indicates automatic formatting |

### Explanation

Sets automatic format function.

When 0 is specified in *val*, it is disabled; when 1 is set, it is enabled.

### Return value

Previously set automatic format value.

### Remarks

This function should be used for testing purposes only.

**See also:**

## _card_chan

Gets a memory card BIOS event.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long_card_chan**(*void*)

### Arguments

None.

### Explanation

Returns the device number of the memory card that just generated an event.

### Return value

2-digit hex device number.

### Remarks

**See also:** card_status (p. 4-12), _card_wait (p. 4-13).

## _card_clear

Clears unconfirmed flags.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long _card_clear**(*chan*)
**long** *chan*;

### Arguments

| | |
|---|---|
| *chan* | Port number x 16 + Card number |

### Explanation

Performs a dummy write to the system management area of the card and clears unconfirmed flags specified in the card.

Port number for Port 1 is zero. Port number for Port 2 is one. Card number is zero when a standard controller is connected. If a multi-tap is connected, then card number may be in the range 0-3.

This function executes asynchronously, so it terminates immediately. Multiplex processing to the same card slot is not performed. That is, multiple _card_clear calls to the same multi-tap cannot be processed synchronously. Actual processing termination is communicated by an event. (See table below.)

**Table 4–1: Posts an event on completion of processing**

| Source Descriptor/Event Class | Contents |
|---|---|
| HwCARD/EvSpIOE | Ends process |
| HwCARD/EvSpTIMOUT | Card not connected |
| HwCARD/EvSpNEW | New card detected |
| HwCARD/EvSpERROR | Error generated |
| HwCARD/EvSpUNKOWN | Source unknown |

### Return value

1 if successful processing registration, otherwise 0.

### Remarks

**See also:** card_info (p. 4-9).

## _card_info

Gets card status.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long _card_info**(*chan*)
**long** *chan*;

### Arguments

| | |
|---|---|
| *chan* | Port number x 16 + Card number |

### Explanation

Tests the connection of the memory card specified in *chan*.

Port number for Port 1 is zero. Port number for Port 2 is one. Card number is zero when a standard controller is connected. If a multi-tap is connected, then card number may be in the range 0-3.

Multiplex processing to the same card slot is not performed. That is, multiple _card_clear calls to the same multi-tap cannot be processed synchronously. Actual processing termination is communicated by an event. (See table below.)

**Table 4–2: Posts an event on completion of processing**

| Source Descriptor/Event Class | Description |
|---|---|
| SwCARD/EvSpIOE | Connected |
| SwCARD/EvSpTIMOUT | Not connected |
| SwCARD/EvSpNEW | No writing after connection |
| SwCARD/EvSpERROR | Generates an error |

### Return value

1 if successful processing registration, otherwise 0.

This function executes asynchronously, so it terminates immediately.

### Remarks

Do not use _new_card() to suppress EvSpNEW.

**See also:** _card_clear (p. 4-8), _new_card (p. 4-15).

## _card_load

Tests logical format

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long _card_load**(*chan*)
**long** *chan*;

### Arguments

| | |
|---|---|
| *chan* | Port number x 16 + Card number |

### Explanation

Reads file management information for the card specified by *chan* in the file system in order to get asynchronous access using the I/O management service.

Port number for Port 1 is zero. Port number for Port 2 is one. Card number is zero when a standard controller is connected. If a multi-tap is connected, then card number may be in the range 0-3.

_card_load must be called at least once before you can use open() on a memory card file in O_NOWAIT mode. The function does not have to be reissued unless a card is changed.This function executes asynchronously, so it terminates immediately. Multiplex processing to the same card slot is not performed. That is, multiple _card_clear calls to the same multi-tap cannot be processed synchronously. Actual processing termination is communicated by an event. (See table below.)

**Table 4–3: Posts an event on completion of processing**

| Source Descriptor/ Event Class | Contents |
|---|---|
| SwCARD/EvSpIOE | Read completed |
| SwCARD/EvSpTIMOUT | Not connected |
| SwCARD/EvSpNEW | Uninitialized card |
| SwCARD/EvSpERROR | Generates an error |

### Return value

1 if the read is successful, otherwise 0.

### Remarks

**See also:** format (see libcd), card_info (p. 4-9).

## _card_read

Reads one block from the memory card.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long _card_read**(*chan*, *block*, **buf*)
**long** *chan*;
**long** *block*;
**long** **buf*;

### Arguments

| | |
|---|---|
| *chan* | Port number x 16 + card number |
| *block* | Target block number |
| *buf* | Pointer to 128 byte data buffer |

### Explanation

Reads 128 bytes of buffer data into *buf* from the target block number (*block*) of the memory card of the specified channel (chan).

Port number for Port 1 is zero. Port number for Port 2 is one. Card number is zero when a standard controller is connected. If a multi-tap is connected, then card number may be in the range 0-3.

This function executes asynchronously so it terminates immediately after completion. Multiplex processing to the same card slot is not performed. Actual processing termination is communicated by an event. (See table below.)

**Table 4–4: Posts an event on completion of processing**

| Source Descriptor / Event Class | Contents |
|---|---|
| HwCARD/EvSpIOE | Ends processing |
| HwCARD/EvSpTIMOUT | Card not connected |
| HwCARD/EvSpNEW | New card detected |
| HwCARD/EvSpERROR | Error generated |
| HwCARD/EvSpUNKOWN | Source unknown |

### Return value

1 if successful processing registration, otherwise 0.

### Remarks

This function exists within the low-level interface and is one of the special functions used for testing.

**See also:** _card_write (p. 4-14), open (see libapi), read (see libapi).

## _card_status

Gets memory card BIOS status.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long _card_status**(*drv*)
**long** *drv*;

### Arguments

| | |
|---|---|
| *drv* | Port number |

### Explanation

Gets the memory card BIOS status of each slot, *drv*. Specify *drv* as 0 for Port 1, 1 for Port 2.

This is a synchronous function.

### Return value

If the memory card BIOS is in run state, it can return any of the following values.

**Table 4–5**

| Value | State |
|---|---|
| 0x01 | Idle processing |
| 0x02 | READ processing |
| 0x04 | WRITE processing |
| 0x08 | Connection test processing registration |
| 0x11 | No registered processing (just prior to EvSpTIMOUT generation) |
| 0x21 | No registered processing (just prior to EvSpERROR generation) |

### Remarks

**See also:** card_wait (p. 4-13), _card_chan (p. 4-7).

## _card_wait

Waits for memory card BIOS completion

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**long _card_status**(*drv*)
**long** *drv*;

### Arguments

| | |
|---|---|
| *drv* | Sets slot number |

### Explanation

Wait until registration processing completes for the *drv* slot. Specify *drv* as 0 for Port 1, 1 for Port 2.

### Return value

Always 1.

### Remarks

**See also:** _card_status (p. 4-12), _card_chan (p. 4-7).

## _card_write

Writes to one block of the memory card.

### Syntax

**long _card_write**(*chan*, *block*, **buf*)
**long** *chan*;
**long** *block*;
**long** **buf*;

### Arguments

| | |
|---|---|
| *chan* | Port number x 16 + card number |
| *block* | Target block number |
| *buf* | Pointer to 128-byte data buffer |

### Explanation

Writes 128 bytes of buffer data pointed to by *buf* to the target block number (*block*) of the memory card of the specified channel (chan).

Specifies Port number x 16 + Card number in *chan*. Port 1 is 0, and Port 2 is 1. The card number is normally 0.

This function executes asynchronously, so it terminates immediately. Multiplex processing to the same card slot is not performed. That is, multiple _card_clear calls to the same multi-tap cannot be processed synchronously. Actual processing termination is communicated by an event. (See table below.)

**Table 4–6: Posts an event on completion of processing**

| Source Descriptor/Event Class | Contents |
|---|---|
| HwCARD/EvSpIOE | Ends process |
| HwCARD/EvSpTIMOUT | Card not connected |
| HwCARD/EvSpNEW | New card detected |
| HwCARD/EvSpERROR | Error generated |
| HwCARD/EvSpUNKOWN | Source unknown |

### Return value

1 if successful processing registration, otherwise 0.

### Remarks

This function exists within the low-level interface and is one of the special functions used for testing only. Do not use this function in your code, it is too low level.

**See also:** _card_read (p. 4-11), open (see libapi), write (see libapi).

## _new_card

Changes settings of unconfirmed flag test.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcard.lib* | *Kernel.h* | *3.0* | *7/31/96* |

### Syntax

**void _new_card**(*void*)

### Arguments

None.

### Explanation

Masks the generation of an EvSpNEW event immediately after _card_read() or _card_write().

Terminates immediately even though it is a synchronous function.

### Return value

None.

### Remarks

**See also:** _card_clear (p. 4-8), _card_read (p. 4-11), _card_write (p. 4-14).

# Chapter 5: Data Processing Library
# Table of Contents

## DECDCTENV

Quantization tables and environment data used during MDEC decoding process.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *3.5* | *7/31/96* |

### Structure

**typedef struct {**
**u_char** *iq_y*[64];
**u_char** *iq_c*[64];
**short** *dct*[64];
**} DECDCTENV;**

### Members

| | |
|---|---|
| *iq_y* | Brightness component quantization table |
| *iq_c* | Chrominance component quantization table |
| *dct* | System reserved |

### Explanation

This structure contains the tables used during the reverse-quantization step of the MDEC decoding process. The default values used by the system are:

**iq_y**

$$\begin{bmatrix} 2 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix} \times 1/16$$

**iq_c**

$$\begin{bmatrix} 2 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix} \times 1/16$$

### Remarks

The values in the *iq_y* and *iq_c* tables are sorted in a diagonal zig-zag scanning order.

## ENCSPUENV*

SPU encode environment attribute structure.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *3.6* | *10/23/96* |

### Structure

```
typedef struct {
        short *src;
        short *dest;
        long size;
        long loop_start;
        char loop;
        char byte_swap;
        char proceed;
        char pad4;
    } ENCSPUENV;
```

### Members

| | |
|---|---|
| *src* | 16-bit PCM data address |
| *dest* | PlayStation original waveform data |
| *size* | 16-bit PCM data size(in bytes) |
| *loop_start* | PCM data loop start point(in bytes) |
| *loop* | Loop waveform generation specification<br>ENCSPU_ENCODE_LOOP:<br>Generate loop waveform data<br>ENCSPU_ENCODE_NO_LOOP:<br>Generate non-loop waveform data |
| *byte_swap* | PCM data endian specification<br>ENCSPU_ENCODE_ENDIAN_BIG:<br>16-bit big endian<br>ENCSPU_ENCODE_ENDIAN_LITTLE:<br>16-bit little endian |
| *proceed* | Whole/Divided encoding specification<br>ENCSPU_ENCODE_WHOLE<br>Whole encoding<br>ENCSPU_ENCODE_START<br>Start divided encoding<br>ENCSPU_ENCODE_CONTINUE<br>Continue divided encoding<br>ENCSPU_ENCODE_END<br>End divided encoding |
| *pad4* | System reserved |

### Explanation

This structure is used to specify the SPU encode environment attributes for EncSPU() function.

### Remarks

When 0 is specified for "loop", "loop_start" will be ignored.

## DecDCTBufSize

Obtains the size of the run-level compressed DCT data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**long DecDCTBufSize** (**bs*)
**unsigned long** **bs*;

### Arguments

| | |
|---|---|
| *bs* | Pointer to bitstream |

### Explanation

This function returns the uncompressed length of the data contained in the Huffman-encoded bitstream pointed to by the *bs* parameter. It does not perform the actual decoding.

### Return value

Length of uncompressed data in long words (i.e. returns 1000 for a 4000-byte length).

### Remarks

**See also:**

## DecDCTGetEnv

Obtain the current quantization tables and environment data used during MDEC image decoding.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *3.5* | *7/31/96* |

### Syntax

**DECDCTENV *DecDCTGetEnv (DECDCTENV** **env***)**

### Argument

| | |
|---|---|
| *env* | Pointer to decoding environment |

### Explanation

This function returns the current decoding environment to *env*.

### Return value

Top address of *env*.

### Remarks

**See also:**

## DecDCTin

Begin decoding of RLE-encoded MDEC image data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**void DecDCTin** (**runlevel*, *mode*)
**unsigned long** **runlevel*;
**long** *mode*;

### Arguments

| | |
|---|---|
| *runlevel* | Pointer to input runlevel |
| *mode* | Decode mode |

### Explanation

Begins decoding the RLE-encoded MDEC image data at the address specified by *runlevel*. A maximum of 128k may be decoded at a time. The resulting image data is retrieved by the DecDCTout() function.

The image depth and transparency is controlled by the *mode* parameter:

**Table 5–1**

| Bit 0 | Output mode |
|---|---|
| 0 | 16-bit direct color |
| 1 | 24-bit direct color |

| Bit 1 | STP |
|---|---|
| 0 | 0 |
| 1 | 1 |

The depth of the output pixels is specified by bit 0; either 24-bit or 16-bit can be selected. If it is 16-bit mode, bit 15 of the pixel (STP bit), can be specified by *mode* bit 1.

### Return value

None.

### Remarks

The image data produced is raw pixel data without any header information of any kind. The width and height of the image produced is not maintained. It is the responsibility of the application or a higher level structure (such as the STR format) to maintain such information.

Data decoded from a single DecDCTin() call may be read using multiple DecDCTout() calls, or the data created by multiple DecDCTin() calls may be read using a single DecDCTout() call.

The DecDCTin() function is non-blocking. To detect when execution of the primitive list is complete, use the DecDCTinSync() function or install a callback routine with the DecDCTinCallback() function. If a DecDCTin() call is executed before a previous one has finished, the transmission will be blocked until the previous operation is complete.

**See also:**

## DecDCTinCallback

Installs a callback routine to be called at termination of MDEC transmission.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**long DecDCTinCallback** (**func*)
**void** (**func*)();

### Arguments

| | |
|---|---|
| *func* | Pointer to callback function address |

### Explanation

This function installs the user-defined callback routine specified by *func*. This routine will be called when the data transmission initiated by a DecDCTin() call has been completed. If *func* is 0, any previous callback routine is disabled.

### Return value

A pointer to a previously set callback function.

### Remarks

Inside the callback, subsequent transmission termination interrupts are masked. Therefore, the callback routine should return as soon as possible. Also note that although the specified function is called during an interrupt, it is not an interrupt handler. It should be written as normal subroutine that will be called by the main interrupt handler.

**See also:**

## DecDCTinSync

Detects DecDCTin() termination.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**long DecDCTinSync** (*mode*)
**long** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Mode |

### Explanation

Detects termination of DecDCTin(). *Mode* values are as follows:

Table 5–2

| Value | Description |
|---|---|
| 0 | Blocks until termination |
| 1 | Performs only status notification |

### Return value

Image processing subsystem status. 1 is returned if transmission is in process and 0 if transmission is not being performed.

### Remarks

**See also:**

## DecDCTout

Receives decoded data from the image processing subsystem.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**void DecDCTout** (**cell*, *size*)
**unsigned long** **cell*;
**long** *size*;

### Arguments

| | |
|---|---|
| *cell* | Pointer to decoded image data |
| *size* | Received data size (long word) |

### Explanation

The RLE-encoded MDEC image data previously specified in a DecDCTin() call is decoded and stored in the buffer specified by the *cell* parameter. The amount of data to be transferred is specified in long words by the *size* parameter (i.e. size=1000 to transfer 4000 bytes of data). Multiple calls to DecDCTout() may be made to retrieve image data.

You must specify a *size* value that is the same as or smaller than the available decoded data. If there is more data available than is read by one DecDCTout() call, then additional calls must be made to avoid MDEC transmission deadlocks.

The decoded image is output one 16 x 16 macroblock at a time. The *size* specified must be a multiple of the total macroblock size for the current decoding mode. If decoding to 16-bit, a macroblock is 128 words. If decoding to 24-bit, the macroblock length is 192 words.

### Return value

None.

### Remarks

The DecDCTout() function is non-blocking. To detect when execution of the primitive list is complete, use the DecDCToutSync() function or install a callback routine with the DecDCToutCallback() function. If a DecDCTout() call is executed before a previous one has finished, the transmission will be blocked until the previous operation is complete.

**See also:**

## DecDCToutCallBack

Installs a callback routine to be called at termination of MDEC transmission.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**long DecDCToutCallback** (**func*)
**long** (**func*)();

### Arguments

| | |
|---|---|
| *func* | Pointer to callback function address |

### Explanation

This function installs the user-defined callback routine specified by *func*. This routine will be called when the data transmission initiated by a DecDCTout() call has been completed. If *func* is 0, and previous callback routine is disabled.

### Return value

A pointer to a previously set callback function.

### Remarks

Inside the callback, subsequent transmission termination interrupts are masked. Therefore, the callback routine should return as soon as possible. Also note that although the specified function is called during an interrupt, it is not an interrupt handler. It should be written as a normal subroutine that will be called by the main interrupt handler.

**See also:**

## DecDCToutSync

Detects termination of DecDCTout().

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**long DecDCToutSync** (*mode*)
**long** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Mode |

### Explanation

Detects termination of DecDCTout(). Mode values are as follows:

**Table 5–3**

| Value | Description |
|---|---|
| 0 | blocks until termination |
| 1 | performs only status notification |

### Return value

Image processing subsystem status. 1 is returned if reception is in progress and 0 if reception is not being performed.

### Remarks

**See also:**

## DecDCTPutEnv

Set image-processing-subsystem environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *3.5* | *7/31/96* |

### Syntax

**DECDCTENV *DecDCTPutEnv (DECDCTENV** **env***)**

### Argument

*env* Pointer to decoding environment

### Explanation

This function sets the quantization tables and environment data used during the reverse-quantization step of the MDEC decoding process.

### Return value

Top address of *env*.

### Remarks

**See also:**

## DecDCTReset

Initializes image processing subsystem.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**void DecDCTReset** (*mode*)
**long** *mode*;

### Arguments

*mode* Reset mode

### Explanation

This function resets the image processing subsystem. Values that can be specified for *mode* are as follows:

Table 5–4

| Value | Content |
|---|---|
| 0 | Initializes all internal states |
| 1 | Discontinues only current decoding; does not affect internal states |

### Return value

None.

### Remarks

Processing time is longer for mode0 than for mode1 because internal tables are initialized.

**See also:**

## DecDCTvlc

Decodes Huffman-compressed MDEC image data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *2.x* | *7/31/96* |

### Syntax

**void DecDCTvlc** (**bs*, **runlevel*)
**unsigned long** **bs*;
**unsigned long** **runlevel*;

### Arguments

| | |
|---|---|
| *bs* | Pointer to bitstream data. Set to NULL to continue processing the previous bitstream. |
| *runlevel* | Pointer to buffer that will receive decompressed data. Set to NULL to continue processing the previous bitstream. |

### Explanation

Decodes the Huffman-compressed MDEC image data pointed to by *bs* and places the resulting data into the *runlevel* buffer. This data must then be passed to DecDCTIn() for the final stage of decompression. Before calling DecDCTvlc(), you can determine the buffer size needed for *runlevel* by using the DecDCTBufSize() function.

### Return value

0 if all data has been decoded, non-0 otherwise.

### Remarks

This is a blocking function.

This function is only the first stage of decoding an MDEC image. The Huffman-encoded bitstream must always be decoded using DecDCTvlc() before DecDCTIn() is executed.

The DecDCTvlcSize() function controls the maximum amount of data decoded by a single call to DecDCTvlc(). When decoding a single bitstream using multiple calls to DecDCTvlc(), the *bs* and *runlevel* parameters should be set to zero on the second and subsequent calls to indicate that you are continuing to process the previous specified bitstream.

**See also:**

## DecDCTvlcSize

Sets the maximum amount of data returned by a single call to the DecDCTvlc() function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *3.2* | *7/31/96* |

### Syntax

**int DecDCTvlcSize (int** *size***)**

### Arguments

*size* Maximum value of a decoded runlevel (long word)

### Explanation

This *size* parameter for the DecDCTvlcSize() function specifies the maximum number of long words that may be returned at once by the DecDCTvlc() function. Subsequent calls to the DecDCTvlc() function will halt after decoding the specified number of long words. If *size* is set to zero, the DecDCTvlc() will decode the entire bitstream regardless of length.

This allows your program to make multiple calls to DecDCTvlc() to decode a bitstream in chunks using a smaller buffer size.

### Return value

Previously set buffer *size*.

**Example:**

```
/* Decoding the first VLC_SIZE word in VLC */
DecDCTvlcSize (VLC_SIZE);
isvlcLeft = DecDCTvlc (next, dec.vlcbuf[dec.vlcid]);
/* Waiting for data to be completed */
do {
    /* Decoding the remaining VLC_SIZE words in VLC */
    if (isvlcLeft) {
        isvlcLeft = DecDCTvlc (0, 0);
        FntPrint ("%d, ", VSync (1));
    }
    /* Application code is here */
} while (isvlcLeft || isEndOfFlame == 0);
isEndOfFlame = 0;
```

### Remarks

This is a block function. A bitstream must be converted to run- levels by DecDCTvlc() before executing DecDCTin().

**See also:**

## EncSPU*

Encodes 16-bit PCM data into PlayStation original waveform format.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libpress.lib* | *Libpress.h* | *3.6* | *10/23/96* |

### Syntax

**long EncSPU** *(***ENCSPUENV** **es_env)*

### Arguments

| | |
|---|---|
| *es_env* | SPU encode environment attribute structure |

### Explanation

This function encodes the PCM data specified in a member "src" of the SPU encode environment attribute structure, "es_env" into the PlayStation original waveform data (VAG, without header information) and returns the encoded data in a member "dest".

Specify the user area address for both members "src" and "dest" of the SPU encode environment attribute structure, "es_env".

Divided encoding can be done by specifying an attribute to a member "proceed" of the SPU encode environment structure, "es_env".

### Return value

ENCSPU_ENCODE_ERROR is returned for a size error of encoded PlayStation original waveform data.

### Remarks

**See also:**

# Chapter 6: Basic Graphics Library
# Table of Contents

## DISPENV

Display environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct DISPENV {**
    **RECT** *disp*;
    **RECT** *screen*;
    **unsigned char** *isinter*;
    **unsigned char** *isrgb24*;
    **unsigned char** *pad0*, *pad1*;
**}**;

### Members

| | |
|---|---|
| *disp* | This is the display area within the frame buffer. Specify the width of the area as one of the following: 256, 320, 368, 512, 640. Specify the area height as 240 or 280. |
| *screen* | Output screen display area. The screen area is calculated without regard to the value of *disp*, using the standard monitor screen upper left-hand point y (0, 0) and lower right-hand point y (256, 240). |
| *isinter* | This is the interlace mode flag.<br>0 non-interlace<br>1 interlace |
| *isrgb24* | This is the 24-bit mode flag.<br>0 16-bit mode<br>1 24-bit mode |
| *pad* | Reserved by system. |

### Explanation

Specifies display parameters for screen display mode, frame buffer display value, and so on.

### Remarks

**See also:**

## DRAWENV

Drawing environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct DRAWENV {**
  **RECT** *clip*;
  **short** *ofs*[*2*];
  **RECT** *tw*;
  **unsigned short** *tpage*;
  **unsigned char** *dtd*;
  **unsigned char** *dfe*;
  **unsigned char** *isbg*;
  **unsigned char** *r0*, *g0*, *b0*;
  **DR_ENV** *dr_env*;
**}**;

### Members

| | |
|---|---|
| *clip* | Drawing area. Drawing is restricted to a short area specified by clip. Drawing is not performed outside the clipping area. (See Remarks 1, below.) |
| *ofs* | Offset. Drawing commands use the added values of (ofs[0], ofs[1]) as an address and draw in the frame buffer. (See Remarks 2.) |
| *tw* | Texture window. The short area texture pattern restricted by the texture page *tw* is used repeatedly. |
| *tpage* | Initial value of texture page |
| *dtd* | Dithering processing flag<br>0: off<br>1: on |
| *dfe* | Drawing to display area flag<br>0: drawing to display area is blocked<br>1: drawing to display area is permitted |
| *isbg* | Drawing area clear flag.<br>0: off<br>1: on<br>Does not clear drawing area when drawing environment is set.<br>Paints entire clip area with brightness values (*r0*, *g0*, *b0*) when drawing environment is set. |
| *r0*, *g0*, *b0* | Background color. Valid only when *isbg* is 1 |
| *dr env* | System reserved |

### Explanation

This sets basic drawing parameters, such as drawing offset and drawing clip area. See the definitions for the DR_MODE and DR_ENV primitives.

### Remarks

*1 Graphics can be actually drawn in an area (0, 0) - (1023, 511) in the graphic space.

*2 The offset value and the address after the addition of the offset are wrapped around at (-1024, -1024) - (1023, 1023).

*3  The values which may be specified for the texture window are restricted to the following combinations:

**Table 6–1**

| *tw*.w, *tw*.x | | | | | |
|---|---|---|---|---|---|
| *tw*.w | 0 (=256) | 16 | 32 | 64 | 128 |
| *tw*.x | 0 | Multiple of 16 | Multiple of 32 | Multiple of 64 | Multiple of 128 |

| *tw*.h, *tw*.y | | | | | |
|---|---|---|---|---|---|
| *tw*.h | 0 (=256) | 16 | 32 | 64 | 128 |
| *tw*.y | 0 | Multiple of 16 | Multiple of 32 | Multiple of 64 | Multiple of 128 |

**See also:**

## DR_AREA

Drawing area change primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Structure

**struct DR_AREA {**
  **unsigned long** **tag*;
  **unsigned long** *code*[2];
**};**

### Members

| | |
|---|---|
| *tag* | Pointer to the next primitive in primitive list |
| *code* | New drawing area information specified by SetDrawArea() function |

### Explanation

The DR_AREA primitive modifies the drawing area of the current drawing environment while a primitive list is being drawn. Use the SetDrawArea() function to set the contents of this primitive.

### Remarks

**See also:**

## DR_ENV

Drawing environment modification primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct DR_ENV {**
  **unsigned long** **tag*;
  **unsigned long** *code*[15];
**};**

### Members

| | |
|---|---|
| *tag* | Pointer to the next primitive in primitive list |
| *code* | New drawing environment information specified by SetDrawEnv() function |

### Explanation

The DR_ENV primitive changes the drawing environment while a primitive list is being drawn. Use the SetDrawEnv() function to specify the new DRAWENV parameters to be used.

### Remarks

This function affects only the drawing environment, not the display environment (see DISPENV for that). The entire drawing environment may be changed using this primitive. See the DRAWENV structure definition for more details. See also the DR_MODE primitive, which sets a subset of the drawing environment.

**See also:**

## DR_LOAD

Load Image primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.4* | *7/31/96* |

### Structure

**typedef struct {**
**unsigned long** **tag*;
**unsigned long** *code[4]*;
**unsigned long** *p[12]*;
**} DR_LOAD;**

### Members

| | |
|---|---|
| *tag* | Pointer to next primitive |
| *code* | Parameters set by SetDrawLoad() |
| *p* | Transfer data |

### Explanation

DR_LOAD transfers data below array *p* to the frame buffer. As with LoadImage () semi-transparence/transparence color control is not carried out. Also, there is no dependence on the drawing environment.

### Remarks

**See also:**

## DR_MODE

Drawing mode modification primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **unsigned long** **tag*;
    **unsigned long** *code*[2];
**} DR_MODE**;

### Members

| | |
|---|---|
| *tag* | Pointer to the next primitive in primitive list |
| *code* | New drawing environment information as specified by SetDrawMode() function |

### Explanation

The DR_MODE primitive changes the texture page, texture window, dithering flag, and drawing flag parameters of the current drawing environment while a primitive list is being drawn. See the *tpage, tw, dtd,* and *dfe* fields of the DRAWENV structure for more information. Use the SetDrawMode() function to specify the parameters to be used.

### Remarks

### See also:

## DR_MOVE

Rectangle domain copy primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.2* | *7/31/96* |

### Structure

**typedef struct {**
    **u_long** *tag*;
    **u_long** *code*;
    **u_long** *code2*;
    **short** *sx*, *sy*;
    **short** *x0*, *y0*;
    **short** *w*, *h*;
**} DR_MOVE;**

### Members

| | |
|---|---|
| *tag* | Hook to the next primitive (reserved) |
| *code* | Primitive ID |
| *sx*, *sy* | Upper left end point of rectangle domain transfer origin |
| *x0*, *y0* | Upper left end point of rectangle domain transfer destination |
| *w*, *h* | Width and height of rectangle domain |

### Explanation

DR_MOVE performs rectangle domain transference. High speed is the same as moveimage().

Unlike the 16-bit SPRT primitive, semi-transparent/transparent color control is not carried out. Also, it is not dependent on the drawing environment.

### Remarks

### See also:

## DR_OFFSET

Drawing offset modification primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **unsigned long** **tag*;
    **unsigned long** *code*[2];
**} DR_OFFSET**;

**Members**

| | |
|---|---|
| *tag* | Pointer to the next primitive in primitive list |
| *code* | New drawing offset information specified by SetDrawOffset() function |

**Explanation**

The DR_OFFSET primitive changes the drawing offset parameters of the current drawing environment while a primitive list is being drawn. See the *ofs* field of the DRAWENV structure for more information. Use the SetDrawOffset() function to specify the parameters to be used.

**Remarks**

**See also:**

## DR_TPAGE

Texture page change primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.5* | *7/31/96* |

**Structure**

```
typedef struct {
    u_long *tag;
    u_long code[2];
} DR_TPAGE;
```

**Members**

| | |
|---|---|
| *tag* | Pointer to the next primitive in primitive list |
| *code* | New texture page information specified by SetDrawTPage() function |

**Explanation**

The DR_TPAGE primitive changes the texture page parameter of the current drawing environment while a primitive list is being drawn. See the *tpage* field of the DRAWENV structure for more information. Use the SetDrawTPage() function to specify the parameters to be used.

**Remarks**

**See also:**

## DR_TWIN

Texture window change primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

**Structure**

```
typedef struct {
    unsigned long *tag;
    unsigned long code[2];
} DR_TWIN;
```

### Members

| | |
|---|---|
| *tag* | Pointer to the next primitive in primitive list |
| *code* | New texture window information specified by SetDrawTexWindow() function |

### Explanation

The DR_TWIN primitive changes the texture window of the current drawing environment while a primitive list is being drawn. See the *tw* field of the DRAWENV structure for more information. Use the SetDrawTexWindow() function to specify the parameters to be used.

### Remarks

**See also:**

## LINE_F2, LINE_F3, LINE_F4

One flat-shaded non-connecting line/ Two flat-shaded connected lines/ Three flat-shaded connected lines.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct LINE_F2 {**
    **unsigned long** **tag*;
    **unsigned char** *r0*, *g0*, *b0*, *code*;
    **short** *x0*, *y0*;
    **short** *x1*, *y1*;
**}**;

**struct LINE_F3 {**
    **unsigned long** **tag*;
    **unsigned char** *r0*, *g0*, *b0*, *code*;
    **short** *x0*, *y0*;
    **short** *x1*, *y1*;
    **short** *x2*, *y2*;
    **unsigned long** *pad*;
**}**;

**struct LINE_F4 {**
    **unsigned long** **tag*;
    **unsigned char** *r0*, *g0*, *b0*, *code*;
    **short** *x0*, *y0*;
    **short** *x1*, *y1*;
    **short** *x2*, *y2*;
    **short** *x3*, *y3*;
    **unsigned long** *pad*;
**}**;

### Member

| | |
|---|---|
| *tag* | Pointer to the next primitive (reserved) |
| *code* | Primitive ID |
| *r0*, *g0*, *b0* | RGB color specifed by straight line |
| *x**, *y** | Coordinate of vertices forming straight line |
| *pad* | Reserved |

**Explanation**

LINE_F2 draws a non-connecting line linking (*x0, y0*) - *(x1, y1)* with the RGB color specifed by (*r0, g0, b0*).

LINE_F3 draws 2 connecting lines linking (*x0, y0*) - (*x1, y1*) - (*x2, y2*) with the RGB color specifed by (*r0, g0, b0*).

LINE_F4 draws 3 connecting lines linking (*x0, y0*) - (*x1, y1*) - (*x2, y2*) - (*x3, y3*), with the RGB color specifed by (*r0, g0, b0*).

**Remarks**

**See also:**

## LINE_G2, LINE_G3, LINE_G4

One Gouraud-shaded non-connecting line/ Two Gouraud-shaded connected lines/ Three Gouraud-shaded connected lines

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Structure**

```
struct LINE_G2 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char r1, g1, b1, p1;
    short x1, y1;
};
```

```
struct LINE_G3 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char r1, g1, b1, p1;
    short x1, y1;
    unsigned char r2, g2, b2, p2;
    short x2, y2;
    unsigned long pad;
};
```

```
struct LINE_G3 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char r1, g1, b1, p1;
    short x1, y1;
    unsigned char r2, g2, b2, p2;
    short x2, y2;
    unsigned char r3, g3, b3, p3;
    short x3, y3;
    unsigned long pad;
};
```

### Members

| | |
|---|---|
| *tag* | Pointer to the next primitive |
| *r0*, *g0*, *b0* | RGB color values |
| *code* | Primitive ID (reserved) |
| *x0*, *y0* | Vertex coordinates |
| *r1*, *g1*, *b1* | RGB color values |
| *p1* | Primitive ID (reserved) |
| *x1*, *y1* | Vertex coordinates |
| *r2*, *g2*, *b2* | RGB color values |
| *p2* | Primitive ID (reserved) |
| *x2*, *y2* | Vertex coordinates |
| *r3*, *g3*, *b3* | RGB color values |
| *p3* | Primitive ID (reserved) |
| *x3*, *y3* | Vertex coordinates |
| *pad* | Reserved |

### Explanation

LINE_G2 draws non-connecting lines linking (*x0*, *y0*) - (*x1*, *y1*) in such a way that their vertices have the RGB color specified by (*r0*, *g0*, *b0*) - (*r1*, *g1*, *b1*), and perform Gouraud shading at the same time.

LINE_G3 draws the connecting lines linking (*x0*, *y0*) - (*x1*, *y1*)- (*x2*, *y2*) in such a way that their vertices have the RGB color specified by (*r0*, *g0*, *b0*) - (*r1*, *g1*, *b1*) - (*r2*, *g2*, *b2*), and perform Gouraud shading at the same time.

LINE_G4 draws connecting lines linking (*x0*, *y0*) - (*x1*, *y1*)- (*x2*, *y2*) - (*x3*, *y3*) in such a way that their vertices have the RGB color specified by (*r0*, *g0*, *b0*) - (*r1*, *g1*, *b1*) - (*r2*, *g2*, *b2*) - (*r3*, *g3*, *b3*) and perform Gouraud shading at the same time.

### Remarks

**See also:**

## POLY_F3, POLY_F4

Flat-shaded, non-textured mapped triangel/ Flat-shaded, not-textured mapped quad.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

```
struct POLY _F3 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    short x1, y1;
    short x2, y2;
};

struct POLY_F4 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    short x1, y1;
    short x2, y2;
    short x3, y3;
```

**};**

### Members

| | |
|---|---|
| *tag* | Pointer to the next primitive |
| *r0*, *g0*, *b0* | RGB color values |
| *code* | Primitive ID (reserved) |
| *x0*, *y0* | Vertex coordinates |
| *x1*, *y1* | Vertex coordinates |
| *x2*, *y2* | Vertex coordinates |
| *x3*, *y3* | Vertex coordinates |

### Explanation

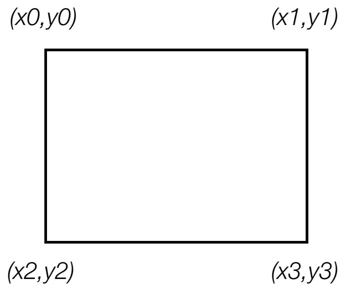
POLY_F3 paints the area demarcated by (*x0*, *y0*) - (*x1*, *y1*) - (*x2*, *y2*) using RGB color specified by (*ro*, *g0*, *b0*).

POLY_F4 paints the area demarcated by (*x0*, *y0*) - (*x1*, *y1*) - (*x3*, *y3*) - (*x2*, *y2*) using RGB color specified by (*ro*, *g0*, *b0*).

The address where a picture is actually drawn is equivalent to the value of *x0-x3* to which the offset value specified by the drawing environment is added. What is drawn is clipped according to the clip area (quadrilateral area) specified by the drawing environment.

Again, if the polygon has a width greater than 1024 and a height greater than 512, all of it will be clipped. In the case of a quadrilateral primitive, the corners are specified in the order shown below.

**Figure 6–1**



### Remarks

**See also:**

## POLY_FT3, POLY_FT4

Flat-shaded, texture-mapped triangle/ Flat-shaded, texture-mapped quad.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct POLY_FT3 {**
    **unsigned long** **tag*;
    **unsigned char** *r0*, *g0*, *b0*, *code*;
    **short** *x0*, *y0*;
    **unsigned char** *u0*, *v0*;
    **unsigned short** *clut*;
    **short** *x1*, *y1*;

```
    unsigned char u1, v1;
    unsigned short tpage;
    short x2, y2;
    unsigned char u2, v2;
    unsigned short pad1;
};

struct POLY_FT4 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char u0, v0;
    unsigned short clut;
    short x1, y1;
    unsigned char u1, v1;
    unsigned short tpage;
    short x2, y2;
    unsigned char u2, v2;
    unsigned short pad1;
    short x3, y3;
    unsigned char u3, v3;
    unsigned short pad2;
};
```

## Members

| | |
|---|---|
| *tag* | Pointer to the next primitive |
| *r0*, *g0*, *b0* | RGB color values |
| *code* | Primitive ID (reserved) |
| *x0*, *y0* | Vertex coordinates |
| *u0*, *v0* | Texture coordinates |
| *clut* | CLUT ID (color-look-up table for 4-bit/8-bit mode only) |
| *x1*, *y1* | Vertex coordinates |
| *u1*, *v1* | Texture coordinates |
| *tpage* | Texture page ID |
| *x2*, *y2* | Vertex coordinates |
| *u2*, *v2* | Texture coordinates |
| *pad1* | Reserved by the system. |
| *x3*, *y3* | Vertex coordinates |
| *u3*, *v3* | Texture coordinates |
| *pad2* | Reserved by the system. |

## Explanation

POLY_FT3 draws an area demarcated by (*x0, y0*) - (*x1, y1*) - (*x2, y2*) while mapping the area demarcated by (u0, v0) - (*u1, v1*) - (*u2, v2*) in the texture pattern on the texture page *tpage*.

POLY_FT4 draws an area demarcated by (*x0, y0*) - (*x1, y1*) - (*x3, y3*) - (*x2, y2*) while mapping the area demarcated by (*u0, v0*) - (*u1, v1*) - (*u3, v3*) - (*u2, v2*) in the texture pattern on the texture page *tpage*.

The actual brightness value for drawn graphics are obtained by multiplying the RGB color values from the texture pattern by the RGB color values given *by r0, g0, b0*.

The texture coordinates are the coordinates (0 to 255) inside the texture page which correspond to the vertices of the triangle to be drawn. if the texture mode is 4-bit or 8-bit, the texture coordinates and the actual frame buffer address will not be 1-to-1.

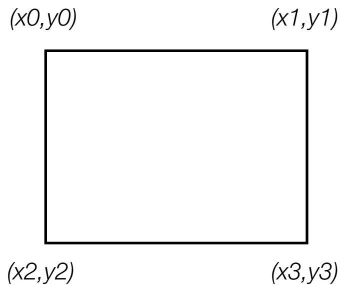
Texture page ID is given to tpage. Using the GetTPage() function, the texture page ID is obtained from the address (*x, y*) of the buffer frame where the texture page is located.

A texture using CLUT gives CLUT ID to be set in clut. Using the GetClut() function, CLUT ID is obtained from the address (*x, y*) of the frame buffer where CLUT is located.

The size of the texture page which can be used by one drawing command is 256 x 256. One primitive can only use one texture page.

In the case of a quadrilateral primitive, the corners are specified in the order shown below. The same applies to designation of (*u*, *v*) for a texture map rectangle, and (*r*, *g*, *b*) for a Gouraud shaded rectangle.

Figure 6–2



### Remarks

**See also:**

## POLY_G3, POLY_G4

Gouraud-shaded, non-textured mapped triangle/ Gourard-shaded, non-textured mapped quad.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

```
struct POLY_G3 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char r1, g1, b1, pad1;
    short x1, y1;
    unsigned char r2, g2, b2, pad2;
    short x2, y2;
};

struct POLY_G4 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char r1, g1, b1, pad1;
    short x1, y1;
    unsigned char r2, g2, b2, pad2;
    short x2, y2;
    unsigned char r3, g3, b3, pad3;
    short x3, y3;
};
```

## Members

| | |
|---|---|
| *tag* | Pointer to the next primitive |
| *r0*, *g0*, *b0* | RGB color values |
| *code* | Primitive ID (reserved) |
| *x0*, *y0* | Vertex coordinates |
| *r1*, *g1*, *b1* | RGB color values |
| *pad1* | Reserved by the system. |
| *x1*, *y1* | Vertex coordinates |
| *r2*, *g2*, *b2* | RGB color values |
| *pad2* | Reserved by the system. |
| *x2*, *y2* | Vertex coordinates |
| *r3*, *g3*, *b3* | RGB color values |
| *pad3* | Reserved by the system. |
| *x3*, *y3* | Vertex coordinates |

## Explanation

When drawing while performing Gouraud shading, POLY_G3 paints the area demarcated by (*x0*, *y0*) - (*x1*, *y1*) - (*x2*, *y2*) so that vertex RGB color value may be set to (*r0*, *g0*, *b0*) - (*r1*, *g1*, *b1*) - (*r2*, *g2*, *b2*).

When drawing while performing Gouraud shading, POLY_G4 paints the area demarcated by (*x0*, *y0*) - (*x1*, *y1*) - (*x3*, *y3*) - (*x2*, *y2*) so that vertex RGB color value may be set to (*r0*, *g0*, *b0*) - (*r1*, *g1*, *b1*) - (*r3*, *g3*, *b3*) - (*r2*, *g2*, *b2*).

The brightness of triangle-internal pixels is calculated by performing linear interpolation of the RGB color values of the three vertices. (Gouraud shading).

## Remarks

**See also:**

## POLY_GT3, POLY_GT4

Gouraud-shaded, texture-mapped triangle/ Gouraud-shaded, texture-mapped quad.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

## Structure

```
struct POLY_GT3 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char u0, v0;
    unsigned short clut;
    unsigned char r1, g1, b1, pad1;
    short x1, y1;
    unsigned char u1, v1;
    unsigned short tpage;
    unsigned char r2, g2, b2, pad2;
    short x2, y2;
    unsigned char u2, v2;
    unsigned char pad2;
};

struct POLY_GT4 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char u0, v0;
    unsigned short clut;
    unsigned char r1, g1, b1, p1;
    short x1, y1;
    unsigned char u1, v1;
    unsigned short tpage;
    unsigned char r2, g2, b2, p2;
    short x2, y2;
    unsigned char u2, v2;
    unsigned char pad2;
    unsigned char r3, g3, b3, p3;
    short x3, y3;
    unsigned char u3, v3;
    unsigned char pad3;
};
```

## Members

| | |
|---|---|
| *tag* | Pointer to the next primitive |
| *r0*, *g0*, *b0* | RGB color values |
| *code* | Primitive ID (reserved) |
| *x0*, *y0* | Vertex coordinates |
| *u0*, *v0* | Texture coordinates |
| *clut* | CLUT ID (color-look-up table for 4-bit/8-bit mode only) |
| *r1*, *g1*, *b1* | RGB color values |
| *pad1* | Reserved by the system. |
| *x1*, *y1* | Vertex coordinates |
| *u1*, *v1* | Texture coordinates |
| *tpage* | Texture page ID |
| *r2*, *g2*, *b2* | RGB color values |
| *pad2* | Reserved by the system. |
| *x2*, *y2* | Vertex coordinates |
| *u2*, *v2* | Texture coordinates |
| *pad3* | Reserved by the system. |
| *r3*, *g3*, *b3* | RGB color values |
| *x3*, *y3* | Vertex coordinates |
| *u3*, *v3* | Texture coordinates |
| *p1* | Primitive ID (reserved) |
| *p2* | Primitive ID (reserved) |
| *p3* | Primitive ID (reserved) |

## Explanation

POLY_GT3 draws a triangle performing texture mapping and Gouraud shading simultaneously.

POLY_GT4 draws a quadrilateral performing texture mapping and Gouraud shading simultaneously.

The actual RGB color values for the picture are equal to the RGB color values obtained from the texture pattern multiplied by the RGB color values calculated by Gouraud shading.

## Remarks

**See also:**

## RECT

Frame buffer rectangular area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct RECT {**
    **short** *x*, *y*;
    **short** *w*, *h*;
**};**

### Members

| | |
|---|---|
| *x*, *y* | Top left coordinates of the rectangular area |
| *w*, *h* | Width and height of the rectangular area |

### Explanation

This structure is used by several library functions to specify a rectangular area of the frame buffer. For these functions, neither negative values, nor values exceeding the size of the frame buffer (1024x512) may be specified.

### Remarks

**See also:**

## SPRT

Sprite of any desired size.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct SPRT {**
    **unsigned long** **tag*;
    **unsigned char** *r0*, *g0*, *b0*, *code*;
    **short** *x0*, *y0*;
    **unsigned char** *u0*, *v0*;
    **unsigned short** *clut*;
    **short** *w*, *h*;
**};**

### Members

| | |
|---|---|
| *tag* | Pointer to next primitive (reserved) |
| *r0*, *g0*, *b0* | RGB color values for sprite |
| *code* | Primitive code (reserved) |
| *x0*, *y0* | Position of sprite (top right coordinate) |
| *u0*, *v0* | Position of sprite texture within the texture page (top right coordinate). *u0* should be an even number. |
| *clut* | CLUT ID used (for 4-bit/8-bit mode only). |
| *w*, *h* | Width and height of sprite. *w* is an even number. |

### Explanation

This draws a texture-mapped rectangular area. Drawing speed for a SPRT primitive is faster than for a POLY_FT4.

### Remarks

Only even numbers can be specified for *u0* and *w*.

Because the SPRT primitive has no *tpage* parameter, the texture page of the current drawing environment is used. Note that you can change the texture page by inserting a DR_TPAGE or DR_MODE primitive into the primitive list to be executed before your SPRT primitive.

**See also:**

## SPRT_8, SPRT_16

8 x 8 fixed size, texture-mapped sprite/ 16 x 16 fixed size, texture-mapped sprite.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

```
struct SPRT_16 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char u0, v0;
    unsigned short clut;
};

struct SPRT_8 {
    unsigned long *tag;
    unsigned char r0, g0, b0, code;
    short x0, y0;
    unsigned char u0, v0;
    unsigned short clut;
};
```

### Members

| | |
|---|---|
| *tag* | Pointer to next primitive (reserved) |
| *r0*, *g0*, *b0* | RGB color values for sprite |
| *code* | Primitive code (reserved) |
| *x0*, *y0* | Position of sprite (top right coordinate) |
| *u0*, *v0* | Position of sprite texture within the texture page (top right coordinate). *u0* should be an even number. |
| *clut* | CLUT ID used (for 4-bit/8-bit mode only). |

### Explanation

This primitive draws a sprite with a fixed size of 8 x 8 or 16 x 16. The same result can be obtained if 8 and 16 are designated as the w and h members for the SPRT structure.

### Remarks

**See also:**

## TILE

Tile Sprite of any desired size.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct TILE {**
  **unsigned long** **tag*;
  **unsigned char** *r0*, *g0*, *b0*, *code*;
  **short** *x0*, *y0*;
  **short** *w*, *h*;
**};**

### Members

| | |
|---|---|
| *tag* | Pointer to next primitive (reserved) |
| *r0*, *g0*, *b0* | RGB color values for sprite |
| *code* | Primitive code (reserved) |
| *x0*, *y0* | Position of sprite (top right coordinate) |
| *w*, *h* | Width and height of sprite. *w* is an even number. |

### Explanation

The rectangular area is drawn with the specified RGB color value (r0, g0, b0). No texture mapping or shading is done. This is faster than the POLY_F4 primitive.

### Remarks

**See also:**

## TILE_1, TILE_8, TILE_16

1 x 1 fixed-size tile sprite/ 8 x 8 fixed-size tile sprite/ 16 x 16 fixed-size tile sprite.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

**struct TILE_16 {**
  **unsigned long** **tag*;
  **unsigned char** *r0*, *g0*, *b0*, *code*;
  **short** *x0*, *y0*;
**};**

**struct TILE_8{**
  **unsigned long** **tag*;
  **unsigned char** *r0*, *g0*, *b0*, *code*;
  **short** *x0*, *y0*;
**};**

**struct TILE_1 {**
  **unsigned long** **tag*;
  **unsigned char** *r0*, *g0*, *b0*, *code*;

```
    short x0, y0;
};
```

## Members

| | |
|---|---|
| *tag* | Pointer to next primitive (reserved) |
| *r0*, *g0*, *b0* | RGB color values for sprite |
| *code* | Primitive code (reserved) |
| *x0*, *y0* | Position of sprite (top right coordinate) |

## Explanation

These primitives are fixed-size versions of the TILE primitive. The rectangular area is drawn with the specified RGB color value (r0, g0, b0). No texture mapping or shading is done. These are faster than the POLY_F4 primitive.

## Remarks

**See also:**

## TIM_IMAGE

TIM format image data header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

## Structure

```
typedef struct {
    unsigned long mode;
    RECT *crect;
    unsigned long *caddr;
    RECT *prect;
    unsigned long *paddr;
} TIM_IMAGE;
```

## Members

| | |
|---|---|
| *mode* | Pixel mode<br>Bits 0-3: Pixel bit depth<br>0: 4-bit CLUT<br>1: 8-bit CLUT<br>2: 15-bit direct<br>3: 24-bit direct<br>4: Mixed<br>Bit 4: CLUT flag<br>0: No CLUT<br>1: Has CLUT |
| *crect* | Pointer to destination rectangle in VRAM for CLUT data |
| *caddr* | Pointer to address of CLUT data in main memory |
| *prect* | Pointer to destination rectangle in VRAM for texture image data |
| *paddr* | Pointer to address of texture image data in main memory |

## Explanation

TIM data header information acquired by the ReadTIM() function.

### Remarks

*crect* and *caddr* are assigned a value of zero for TIM having no CLUT.

**See also:**

## TMD_PRIM

TMD format model data header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Structure

```
typedef struct {
    unsigned long  id;
    unsigned char  r0, g0, b0, p0;
    unsigned char r1, g1, b1, p1;
    unsigned char  r2, g2, b2, p2;
    unsigned char  r3, g3, b3, p3;
    unsigned short  tpage, clut;
    unsigned char  u0, v0, u1, v1;
    unsigned char  u2, v2, u3, v3;
    SVECTOR x0, x1, x2, x3;
    SVECTOR n0, n1, n2, n3;
    SVECTOR *v_ofs;
    SVECTOR *n_ofs;
    unsigned short  vert0, vert1;
    unsigned short vert2, vert3;
    unsigned short norm0, norm1;
    unsigned short norm2, norm3;
} TMD_PRIM;
```

### Members

| | |
|---|---|
| *id* | TMD primitive ID |
| *r0*, *g0*, *b0*,...*r3*, *g3*, *b3* | RGB color values of the vertices of a primitive |
| *clut* | CLUT ID used by a primitive |
| *tpage* | Texture page used by a primitive |
| *u0*, *v0*, *u1*, *v1*..*u3*, *v3* | Texture coordinates of the vertices of a primitive |
| *x0*, *x1*, *x2*, *x3* | Three-dimensional coordinates of a primitive |
| *n0*, *n1*, *n2*, *n3* | Normal coordinates of a primitive |
| *v_ofs* | Pointer to start coordinates of a vertex array |
| *n_ofs* | Pointer to start coordinates of a normal array |
| *vert0*, *vert1*..*vert3* | Offset to a vertex array |
| *norm0*, *norm1*..*norm3* | Offset to a vertex array |

### Explanation

Information on primitives constituting a TMD object. The information is acquired using the ReadTMD() function. *x0*, *x1*, *x3*, *n0*, *n1*,*n3* are used for an independent vertex model. *v_ofs*, *n_ofs* and *vert0*,..*vert3*, *norm0*...*norm3* are used for a common vertex model.

### Remarks

Some members have no meaning depending on the TMD primitive type.

**See also:**

## AddPrim

Adds a primitive to a linked list of primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void AddPrim** (**ot*, **p*)
**unsigned long** **ot*;
**unsigned long** **p*;

### Arguments

| | |
|---|---|
| *ot* | Pointer to a primitive or an entry in an ordering table array |
| *p* | Pointer to a primitive |

### Explanation

This function registers a primitive beginning with the address **p* to the OT entry **ot* in OT table. *ot* is an ordering table or pointer to another primitive.

### Return value

None.

### Remarks

A primitive may only be added to a primitive list once. Attempting to add it multiple times will result in a corrupted list.

**See also:**

## AddPrims

Inserts one linked list of primitives into another linked list of primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void AddPrims** (**ot*, **ps*, **pe*)
**unsigned long** **ot*;
**unsigned long****ps*;
**unsigned long** **pe*;

### Arguments

| | |
|---|---|
| *ot* | Pointer to a primitive or an entry in an ordering table array |
| *ps* | Pointer to first primitive in source list |
| *pe* | Pointer to last primitive in source list |

### Explanation

This function inserts one linked list of primitives into another. The *ot* parameter is a pointer to either a primitive or an entry in an ordering table array. This is where the second list will be inserted. The *ps* and *pe* parameters are pointers to the first and last items of the second list. What happens is that the current tag pointer of the item at *ot* will be copied to *pe* and then replaced with a pointer to *ps.*

### Return value

None.

### Remarks

**See also:**

## addVector

Adds vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**addVector** (*v0*, *v1*)

### Arguments

| | |
|---|---|
| *v0*, *v1* | Pointers to vectors |

### Explanation

This macro adds *v1* to the vector v0, and stores the result in *v0*.

### Return value

None.

### Remarks

addVector() is a macro, so there is no dependence on the vector type.

**See also:**

## applyVector

Adds vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.4* | *7/31/96* |

### Syntax

**applyVector** (**v*, *x*, *y*, *z*, *op*)

### Arguments

| | |
|---|---|
| *v* | Pointer to vector |
| *x*,*y*,*z* | Coordinate value |
| *op* | Operator |

### Explanation

Performing the operation specified on vector v, x, y, z and *op*

```
applVector (v, 2, 4, 8, +=)
```

is equivalent to:

v->vx += 2, v-> vx+= 4, v-> vx += 8

### Return value

None.

### Remarks

applyVector is a macro, so there is no dependence on the vector model.

**See also:**

## BreakDraw

Interrupts drawing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.4* | *7/31/96* |

### Syntax

**u_long *BreakDraw** *(void)*

### Arguments

None.

### Explanation

When issued during drawing, the drawing of the polygon presently being drawn will be interrupted after completion. Because the entry of the next polygon drawing returns, if the DrawOTag () return value is issued in the argument, redrawing is possible.

### Return value

Next polygon drawing entry.

### Remarks

**See also:**

## CatPrim

Concatenates a primitive list.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long** **CatPrim* (**p0*, **p1*)
**unsigned long** **p0*, **p1*;

### Arguments

| | |
|---|---|
| *p0*, *pl* | Pointer to start addresses of primitive to be concatenated |

### Explanation

This function links the primitive *p1* to the primitive *p0*.

### Return value

Start address of *p0*.

### Remarks

AddPrim() adds a primitive to a primitive list. CatPrim() simply concatenates two primitives.

**See also:**

## CheckPrim

Checks the validity of the specified primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**long CheckPrim** (**s*, **p*)
**char** **s*;
**unsigned long** **p*;

### Arguments

| | |
|---|---|
| *s* | Pointer to optimal character string |
| *p* | Pointer to primitive start address |

### Explanation

This function checks the validity of the primitive. If the primitive is found to be invalid, it prints a message with the contents of the *s* parameter followed by the type code and length of the primitive. The primitive is not modified in any case.

### Return value

Returns 0 for valid primitive. Returns -1 for an invalid primitive.

### Remarks

**See also:**

## ClearImage

Clears Frame Buffer at high speed.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**int ClearImage** (**recp*, *r*, *g*, *b*)
**RECT** **recp*;
**unsigned char** *r*, *g*, *b*;

### Arguments

| | |
|---|---|
| *recp* | Pointer to rectangular area to be cleared |
| *r*, *g*, *b* | Pixel values to be used for clearing |

### Explanation

Clears a rectangular area inside the Frame Buffer specified by *recp* at RGB color values indicated by (*r*, *g*, *b*).

### Return value

Number in the queue

### Remarks

Because this is a non-blocking function, the end of actual transfer must be detected using DrawSync(). The drawing area will not be affected by the drawing environment (clip/offset).

**See also:**

## ClearOTag

Initializes an array to a linked list for use as an ordering table.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long *ClearOTag** (**ot*, *n*)
**unsigned long** **ot*;
**int** *n*;

### Arguments

| | |
|---|---|
| *ot* | OT starting pointer |
| *n* | Number of entries in OT |

### Explanation

This function walks the array specified by the *ot* parameter and sets each element to be a pointer to the following element, except the last. The *n* parameter specifies how many entries are present in the array. The last element of the array is set to a pointer to a special terminator value which the PlayStation uses to recognize the end of a primitive list.

### Return value

None.

### Remarks

When you want to execute the OT initialized by “ClearOTag()”, execute “DrawOTag (ot)”.

**See also:**

## ClearOTagR

Initializes an array to a linked list for use as an ordering table.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void ClearOTagR** (**ot*, *n*)

**unsigned long** **ot*;
**long** *n*;

### Arguments

| | |
|---|---|
| *ot* | Head pointer of OT |
| *n* | Number of entries in OT |

### Explanation

This function walks the array specified by the *ot* parameter and sets each element to be a pointer to the previous element, except the first. The *n* parameter specifies how many entries are present in the array. The first element of the array is set to a pointer to a special terminator value which the PlayStation uses to recognize the end of a primitive list.

### Return value

None.

### Remarks

When you want to execute the OT initialized by“ClearOTagR()”, execute “DrawOTag (ot+n-1)”.

**See also:**

## copyVector

Copies vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**copyVector** (*vo*, *v1*)

### Arguments

| | |
|---|---|
| *vo*, *v1* | Vector pointer |

### Explanation

Copies vector *v0* to *v1*.

### Return value

### Remarks

copyVector is a macro, so there is no dependence on the vector type.

**See also:**

## DrawPrim

Draws primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Syntax**

**void DrawPrim** (**void** **p* )

**Arguments**

| | |
|---|---|
| *p* | Pointer to primitive. |

**Explanation**

Executes a primitive which has completed initialization.

**Return value**

None.

**Remarks**

Since DrawPrim() is a non-block function, it is necessary to detect the actual end of the transfer on DrawSync(). Slower speed than DrawOTag().

**See also:**

## DrawOTag

Executes a list of GPU primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

**Syntax**

**void DrawOTag** (**ot*)
**unsigned long** **ot*;

**Arguments**

| | |
|---|---|
| *ot* | Pointer to a linked list of GPU primitives |

**Explanation**

This function executes the GPU primitives in the specified link list.

**Return value**

**Remarks**

The DrawOTag() function is non-blocking. To detect when execution of the primitive list is complete, use the DrawSync() function or install a callback routine with the DrawSyncCallback() function.

**See also:**

## DrawOTagEnv

Executes a list of GPU primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

#### Syntax
**void DrawOTagEnv** (**u_long** **p*, **DRAWENV** **env*)

#### Arguments

| | |
|---|---|
| *p* | OT start pointer |
| *env* | Drawing environment |

#### Explanation
Sets the basic parameters for the drawing such as drawing offset/drawing clip area and collectively executes the primitives registered on OT.

#### Return value
None.

#### Remarks
Following the drawing environment specified by DrawOTagEnv, PutDrawEnv ( ) or DrawOTagEnv ( ) will be executed or will be effective until the DR_ENV primitive is executed.

**See also:**

## DrawOTagIO

Sets the drawing environment and draws the primitive registered on OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

#### Syntax
**void DrawOTagIO** (**u_long** **p*)

#### Arguments

| | |
|---|---|
| *p* | Pointer to top of OT |

#### Explanation
Collectively executes the primitives registered on OT.

#### Return value
None.

#### Remarks
Despite the fact that DrawOTagIO does not carry out the same operation as DrawOTag in checking the primitive adjustability, this is executed by means of the CPU. For debugging use.

**See also:**

## DrawSync

Wait for all drawing to terminate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**long DrawSync** (*mode*)
**long** *mode*;

### Arguments

The values which can be specified for mode are shown below.

Table 6–2

| Value | Content |
|---|---|
| 0 | Wait for the termination of all non-block functions registered in the queue. |
| 1 | Find out and return the number of positions in the current queue. |

### Explanation

This function waits for drawing to terminate.

### Return value

The Return value is the number of positions in the execution queue.

### Remarks

If DrawSync(0) is used, and execution of the primitive list takes an exceptionally long time (approximately longer than 8 Vsync) to complete, a timeout is generated and the GPU is reset. Reasons why this might occur include an exceptionally long primitive list, or one that renders exceptionally large numbers of pixels. Another possibility is that the primitive list has been corrupted in some way. To avoid this, the application can use a loop such as:

```
while(DrawSync(1));
```

**See also:**

## DrawSyncCallback

Defines a callback function to be called when the GPU is finished executing a primitive list.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void DrawSyncCallback** (**func*)
**void** (**func*)();

### Arguments

| | |
|---|---|
| *func* | Pointer to callback function |

### Explanation

This defines the routine to be used as a callback when drawing is completed. When all requests in the queue have terminated, the function *func* is called. If *func* is set to 0, then any previous callback routine is disabled.

### Return value

None.

### Remarks

Inside the callback, subsequent drawing termination interrupts are masked. Therefore, the callback routine should return as soon as possible. Also note that although the specified function is called during an interrupt, it is not an interrupt handler. It should be written as a normal subroutine that will be called by the main interrupt handler.

**See also:**

## DumpClut

Printing contents of “clut” member of primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void DumpClut** (*clut*)
**unsigned short** *clut*;

### Arguments

| | |
|---|---|
| *clut* | CLUT ID |

### Explanation

This function prints the CLUT ID contents.

### Return value

None.

### Remarks

**See also:**

## DumpDispEnv

Printing contents of display environment Structure.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void DumpDispEnv** (**env*)
**DISPENV** **env*;

### Arguments

| | |
|---|---|
| *env* | Pointer to display environment |

### Explanation

This function prints the contents of the display environment structure.

### Return value

None.

### Remarks

### See also:

## DumpDrawEnv

Printing contents of drawing environment Structure.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void DumpDrawEnv** (**env*)
**DRAWENV** **env*;

### Arguments

| | |
|---|---|
| *env* | Pointer to drawing environment |

### Explanation

This function prints the contents of the drawing environment structure.

### Return value

None.

### Remarks

### See also:

## DumpOTag

Prints the primitives registered in OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void DumpOTag** (**ot*)
**unsigned long** **ot*;

### Arguments

| | |
|---|---|
| *ot* | OT starting pointer |

### Explanation

This function prints the code field of the primitives registered in the OT.

### Return value

None.

### Remarks

**See also:**

## DumpTPage

Prints the contents of “tpage” member of primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void DumpTPage** (*tpage*)
**unsiged short** *tpage*;

### Arguments

| | |
|---|---|
| *tpage* | texture page ID |

### Explanation

This function prints the contents of the texture page ID.

### Return value

None.

### Remarks

**See also:**

## FntFlush

Draws contents of print stream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long** ***FntFlush** (*id*)
**long** *id*;

### Arguments

| | |
|---|---|
| *id* | Print stream ID |

### Explanation

This function draws the contents of the print stream into the frame buffer. It initializes and then draws a sprite primitive list corresponding to the characters specified in the print stream.

### Return value

The return value is the starting pointer of the primitive list used to perform the drawing.

### Remarks

After the drawing has been done, the print stream contents are also flushed.

**See also:**

## FntLoad

Transmits font pattern.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void FntLoad** (*tx*, *ty*)
**long** *tx*, *ty*

### Arguments

| | |
|---|---|
| *tx*, *ty* | Font pattern frame buffer address |

### Explanation

This function transmits the built-in text font used for debugging text output to the frame buffer. It loads the basic font pattern (4-bit, 256x128) and initializes all the print streams.

### Return value

None.

### Remarks

FntLoad() must always be executed before FntOpen() and FntFlush(). The font area must not clash with the frame buffer area used by the application.

**See also:**

## FntOpen

Opens a print stream for printing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**long FntOpen** (*x*, *y*, *w*, *h*, *isbg*, *n*)
**long** *x*, *y*;
**long** *w*, *h*;
**long** *isbg*;
**long** *n*;

### Arguments

| | |
|---|---|
| *x*, *y* | Display start location |
| *w*, *h* | Display area |
| *isbg* | Automatic clearing of background<br>0: Clear background to (0, 0, 0) when display is performed<br>1: Do not clear background to (0, 0, 0) when display is performed. |
| *n* | Maximum number of characters |

### Explanation

This function opens the stream for on-screen printing. After this, character strings up to *n* characters long can be drawn in the (*x*, *y*)- (*x*+*w*, *y*+*h*) rectangular area of the frame buffer, using FntPrint(). If “1” is specified for isbg, the background is cleared when a character string is drawn.

### Return value

The return value is the stream ID.

### Remarks

Up to 4 streams can be opened at once. However, once a stream is opened, it cannot be closed until the next time FntLoad() is called.

**See also:**

## FntPrint

Prints the specified string to an open print stream, using the same arguments and formatting parameters as the C library printf() function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**long FntPrint** (*id*, **format*, [arg]...)
**long** *id*;
**char** **format*;

### Arguments

| | |
|---|---|
| *id* | Print stream ID |
| *format* | Pointer to print format |

### Explanation

This function sends the string *format* to the specified print stream using the same interface as the fprintf() standard C library function.

### Return value

The return value is the number of characters in the stream.

### Remarks

The character string is not actually displayed until FntFlush() has been executed.

**See also:**

## GetClut

Calculating the value of the “CLUT” member in a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned short GetClut** (*x*, *y*)
**long** *x*, *y*;

### Arguments

| | |
|---|---|
| *x*, *y* | Frame buffer address of CLUT |

**Explanation**

This function calculates and returns the texture CLUT ID.

**Return value**

CLUT ID

**Remarks**

The CLUT address is limited to multiples of 64 in the x direction.

**See also:**

## GetDispEnv

Gets the current display environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Syntax**

**DISPENV** ***GetDispEnv** (**env*)
**DISPENV** **env*;

**Arguments**

| | |
|---|---|
| *env* | Pointer to display environment start address |

**Explanation**

This function stores the current display environment in the address specified by env.

**Return value**

The return value is a pointer to the display environment obtained by the function.

**Remarks**

**See also:**

## GetDrawEnv

Gets the current drawing environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Syntax**

**DRAWENV** ***GetDrawEnv** (**env*)
**DRAWENV** **env*;

**Arguments**

| | |
|---|---|
| *env* | Pointer to drawing environment start address |

**Explanation**

This function stores the current drawing environment in the address specified by *env*.

### Return value
The return value is a pointer to the drawing environment obtained.

### Remarks

**See also:**

## GetGraphDebug

Gets present debug level.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax
**int GetGraphDebug (void)**

### Arguments
None.

### Explanation
Gets graphics system debug level.

### Return value
Present debug level value.

### Remarks

**See also:**

## GetTimSize

Calculates the size of the Tim data domain returned by Krom2Tim ().

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax
**int GetTimSize** (**u_char** **sjis*)

### Arguments
*sjis* Pointer to sjis character string

### Explanation
Calculates size of the Tim data domain returned by Krom2Tim (). This size domain is maintained in malloc ( ) and is designated Krom2Tim ().

### Return value
Size of Tim data domain returned by Krom2Tim ().

### Remarks

### See also:

## GetTPage

Calculates the value of the member “tpage” in a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned short GetTPage** (*tp*, *abr*, *x*, *y*)
**long** *tp*, abr, x, y;

### Arguments

| | |
|---|---|
| *tp* | Texture mode<br>0: 4bitCLUT<br>1: 8bitCLUT<br>2: 16bitDirect |
| *abr* | Semi-transparency rate<br>0: 0.5 x Back + 0.5 x Forward<br>1: 1.0 x Back + 1.0 x Forward<br>2: 1.0 x Back - 1.0 x Forward<br>3: 1.0 x Back + 0.25 x Forward |
| *x*, *y* | Texture page address |

### Explanation

This function calculates the texture page ID, and returns it.

The texture page address is limited to a multiple of 64 in the X direction and a multiple of 256 in the Y direction.

The values that may be specified for *tp* and abr are as follows.

**Table 6–3**

| *tp* | Content |
|---|---|
| 0 | 4-bit CLUT |
| 1 | 8-bit CLUT |
| 2 | 16-bit Direct |

| *abr* | Content |
|---|---|
| 0 | 0.5 x Back + 0.5 x Forward |
| 1 | 1.0 x Back + 1.0 x Forward |
| 2 | 1.0 x Back - 1.0 x Forward |
| 3 | 1.0 x Back + 0.25 x Forward |

### Return value

Texture page ID.

### Remarks

The semitransparent rate is also effective for polygons on which texture mapping is not performed.

**See also:**

## GetVideoMode

Obtains present video signaling system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *3.1* | *7/31/96* |

### Syntax

**long GetVideoMode** *(void)*

### Arguments

None.

### Explanation

Returns the present video signaling system declared in SetVideoMode.

### Return value

Video signaling system mode

**Table 6–1**

| Return Value | Contents |
|---|---|
| MODE_NTSC | NTSC system video signaling system |
| MODE_PAL | PAL system video signaling system |

### Remarks

When SetVideoMode () is not called, no matter what the machine, it will return MODE_NTSC.

**See also:**

## IsEndPrim

Decides the final ending primitive of the list.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**int IsEndPrim** (**void** **p*)

### Argument

| | |
|---|---|
| *p* | Primitive start address |

### Explanation

Decides if the end of the primitive list is *p.*

### Return value

Returns 1 in final end case and returns 0 in non-final end case.

### Remarks

**See also:**

## IsIdleGPU*

Checks if the drawing once suspended by BreakDraw was completed.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.6* | *10/23/96* |

### Syntax

**int IsIdleGPU** (**int** *maxcount)*

### Argument

*maxcount* Count value

### Explanation

Although drawing is suspended by BreakDraw, GPU will not stop until the drawing is completed. Thus this function checks if the drawing suspended by BreakDrawhas been completed or not. If GPU will not be an idle state within the time given by maxcount, -1 will be returned.

### Return value

0: GPU is in idle state. -1: GPU is in drawing state.

### Remarks

**See also:**

## KanjiFntClose

Closes the printstream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax

**void KanjiFntClose (void)**

### Argument

None.

### Explanation

This function closes all the streams currently open and are used by KanjiFntPrint() and initialize the state.

### Return value

None.

### Remarks

Since KanjiFntClose() only initializes the internal state, this function operates even when there is no stream opened at the invocation of the function.

**See also:**

## KanjiFntFlush

Draws contents of the specified Kanji print stream.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long *KanjiFntFlush(***id***)**
**int** *id*;

### Argument

| | |
|---|---|
| *id* | Print stream ID |

### Explanation

This function draws the contents of the Kanji print stream into the frame buffer. It initializes and then draws a sprite primitive list corresponding to the characters specified in the print stream.

### Return value

Start pointer of a primitive list used for drawing

### Remarks

The contents of a print stream are also flushed after the end of drawing.

**See also:**

## KanjiFntOpen

Opens a print stream for printing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax

**int KanjiFntOpen**(*x*, *y*, *w*, *h*, *dx*, *dy*, *cx*, *cy*, *isbg*, *n*)
**int** *x*, *y*, *w*, *h*, *dx*, *dy*, *cx*, *cy*, *isbg*, *n*;

### Arguments

| | |
|---|---|
| *x*, *y* | Position of starting display |
| *w*, *h* | Display area |
| *dx*, *dy* | Kanji font pattern frame buffer address |
| *cx*, *cy* | Kanji clut frame buffer address |
| *isbg* | Automatic background clear |
| | 0 Clears the background to (0, 0, 0) during display. |
| | 1 Does not clear the background to (0, 0, 0) during display. |
| *n* | Maximum number of characters |

### Explanation

This function opens a stream for open screen print. Then, the KanjiFntPrint() function can be used to render a character string composed of up to *n* characters in the rectangular area of (*x*, *y*) and (*x*+*w*, *y*+*h*) on the frame buffer. With *isbg* assigned a value of one, the background is cleared when a character string is rendered.

### Return value

Stream ID.

### Remarks

Up to eight streams can be opened at a time. The opened stream cannot be closed until the KanjiFntLoad() function is called.  The kanji font area must not interfere in the frame buffer area used for applications.

**See also:**

## KanjiFntPrint

Prints the specified string, in SJIS ZENKAKU format, to an open print stream, using the same arguments and formatting parameters as the C library printf() function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax

**int KanjiFntPrint** (*id*, **format*, [arg]...)
**int** *id*
**char** **format*

### Arguments

| | |
|---|---|
| *id* | Print stream ID |
| *format* | Pointer to print format |

### Explanation

Send SJIS ZENKAKU string using printf() interface.

### Return value

Number of characters within the stream.

### Remarks

KANJI code must be the SJIS. Although both ZENKAKU and HANKAKU characters can be mixed in the string, a HANKAKU character will be converted to ZENKAKU when it is drawn. HANKAKU KANA characters are not supported. Actual drawing of the string will be done at execution of KanjiFntFlush(). When there is ~p in the string format, all the characters after ~p will be drawn in half-pitch.

**See also:**

## Krom2Tim

Converts SJIS character string to 4 bits clut Tim data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax

**int Krom2Tim (**
**u_char** **sjis*;
**u_long** **taddr*;
**int** *dx*;
**int** *dy*;

**int** *cdx*;
**int** *cdy*;
**u_int** *fg*;
**u_int** *bg*;
**)**

### Arguments

| | |
|---|---|
| *sjis* | SJIS character string |
| *taddr* | Tim area for storing data |
| *dx, dy* | Pixel data x,y coordinates on VRAM |
| *cdx, cdy* | Clut data x,y coordinates on VRAM |
| *fg, bg* | Character color and bg color |

### Explanation

Converts SJIS character string to 4 bits clut TIM data and returns to addr.

### Return value

When an abnormal code is given, -1 is returned.

### Remarks

The size area returned by GetTimSize must be secured in advance.

The Kanji code must be SJIS. Full-width and half-width characters can be mixed within the character string, but when they are displayed, they will all be converted to full-width characters. Half-width characters are not supported.

**See also:**

## LoadClut

Loads texture CLUT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned short LoadClut** (**col*, *x*, *y*)
**unsigned long** **col*;
**long** *x*, *y*;

### Arguments

| | |
|---|---|
| *col* | Pointer to CLUT data start address |
| *x*, *y* | Destination coordinates in frame buffer |

### Explanation

This function loads 256 entries of texture color data (CLUT) from main memory address *clut* into the frame buffer area starting at coordinate (x,y) and calculates the ID of the loaded texture CLUT.

### Return value

The Return value is the CLUT ID for the loaded CLUT.

### Remarks

256 palette entries are always transmitted, even in 4-bit mode.

**See also:**

## LoadClut2

Loads CLUT for 16 colors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax

**u_short LoadClut2** (**u_long** **clut*, **int** *x*, **int** *y*)

### Arguments

| | |
|---|---|
| *clut* | Pointer to CLUT data start address |
| *x*, *y* | Destination coordinates in frame buffer |

### Explanation

This function loads 16 entries of texture color data (CLUT) from main memory address *clut* into the frame buffer area starting at coordinate (x,y) and calculates the ID of the loaded texture CLUT.

### Return value

CLUT ID for loaded CLUT.

### Remarks

This function transfers the CLUT data of only 16 colors.

**See also:**

## LoadImage

Transfers data to a frame buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void LoadImage** (**recp*, **p*)
**RECT** **recp*;
**unsigned long** **p*;

### Arguments

| | |
|---|---|
| *recp* | Pointer to destination rectangular area |
| *p* | Pointer to main memory address of source of transmission |

### Explanation

This function transfers the contents of memory from the address *p* to the rectangular area in the frame buffer specified by *recp*.

### Return value

None.

### Remarks

Because LoadImage() is a non-block function, the transmission termination has to be detected by “DrawSync()”.

The transfer areas at the source and destination are not affected by the drawing environment (clip, offset). The destination area must be located within a drawable area (0, 0) - (1023, 511). See the description of the DR_LOAD primitive.

**See also:**

## LoadTPage

Loads a texture page.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned short LoadTPage** (**pix*, *tp*, *abr*, *x*, *y*, *w*, *h*)
**unsigned long** **pix*;
**int** *tp*, *abr*, *x*, *y*, *w*, *h*;

### Arguments

| | |
|---|---|
| *pix* | Pointer to texture pattern start address |
| *tp* | Transfer texture type |
| *abr* | Semi-transparency rate |
| *x*, *y* | Destination frame buffer address |
| *w*, *h* | Texture pattern size |

### Explanation

This function loads a texture pattern from the memory area starting at the address *pix* into the frame buffer area starting at the address (*x*, *y*), and calculates the texture page ID for the loaded texture pattern.

### Return value

Texture page ID for the loaded texture pattern.

### Remarks

The texture pattern size is not the actual size of the transfer area in the frame buffer. The texture pattern size is net in pixels.

LoadTPage() starts from within LoadImage().

**See also:**

## MargePrim

Unites number primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *3.0* | *7/31/96* |

### Syntax

**int MargePrim** (**void** **p0*, **void** **p1*)

### Arguments

| | |
|---|---|
| *p0* | Primitives that are connected |
| *p1* | Primitives that connect |

#### Explanation

Links primitive *p0* to primitive *p1.* All following linked primitives are, as usual, able to process AddPrim ().

#### Return value

If successful, returns 0, in cases of failure returns -1.

#### Remarks

*p0* and *p1* are essential to the linked memory domain.

The combined primitive size of p0 and p1 must be less than 15 words.

**See also:**

## MoveImage

Transfers data between two locations within the frame buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

#### Syntax

**int MoveImage** (**rect*, *x*, *y*)
**RECT** **rect*;
**int** *x*, *y*;

#### Arguments

| | |
|---|---|
| *rect* | Pointer to source rectangular area |
| *x*, *y* | Top left corner of the destination rectangle |

#### Explanation

The rectangular area of the frame buffer specified by *rect* is transmitted to the rectangular area of the same size which starts at (*x*, *y*).

The content at the source is preserved. If the source and destination areas are the same, normal operation is not guaranteed.

#### Return value

Number in the queue.

#### Remarks

Because MoveImage() is a non-block function, the termination of the transmission has to be detected by DrawSync().

The transfer areas at the source and destination are not affected by the drawing environment (clip, offset). The destination area must be located within a drawable area (0, 0) - (1023, 511). See also the description of the DR_MOVE primitive.

**See also:**

## NextPrim

Returns pointer to next primitive in primitive list.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long *NextPrim**(**p*)
**unsigned long** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to start address of a primitive |

### Explanation

This function returns a pointer to the next primitive in a primitive list.

### Return value

Pointer to the next primitive.

### Remarks

**See also:**

## OpenTIM

Opens TIM data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**long OpenTIM** (**addr*)
**unsigned long** **addr*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to main memory address to which the TIM has been loaded |

### Explanation

This function opens a TIM in main memory. The information in the opened TIM can then be read using the ReadTIM() function.

### Return value

If it succeeds, "0" is returned. Any other value indicates failure.

### Remarks

Only one TIM can be opened at a time. An opened TIM is not closed until the next time OpenTIM() is called.

**See also:**

## OpenTMD

Opens TMD data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**long OpenTMD** (**tmd*, *obj_no*)
**unsigned long** **tmd*;
**long** *obj_no*;

### Arguments

| | |
|---|---|
| *tmd* | Pointer to main memory address to which TMD has been loaded |
| *obj_no* | Object number. |

### Explanation

This function opens the TMD of the object specified by the *obj_no* parameter. The information in the opened TMD can then be read using the ReadTMD() function.

### Return value

Returns the number of polygons comprising the object as a positive integer. Returns a negative number if it fails.

### Remarks

Calling OpenTMD() closes any previously opened TMD.

**See also:**

## PutDispEnv

Sets the display environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**DISPENV** ***PutDispEnv** (**env*)
**DISPENV** **env*;

### Arguments

*env* Pointer to display environment start address

### Explanation

This function sets a display environment according to information specified by *env*. The display environment is executed as soon as the function is called.

### Return value

This is a pointer to the display environment which has been set. (If the setting failed, the Return value is “0”.)

### Remarks

**See also:**

## PutDrawEnv

Sets the drawing environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|

| | | | |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**DRAWENV** ***PutDrawEnv*** (**env*)
**DRAWENV** **env*;

### Arguments

*env* Pointer to drawing environment start address

### Explanation

Basic drawing parameters such as the drawing offset and the drawing clip area should be set in accordance with the setting specified in *env*.

### Return value

This is a pointer to the drawing environment which has been set. (If setting failed, the Return value is “0”.)

### Remarks

The drawing environment specified using “PutDrawEnv()” is effective until the next time “PutDrawEnv()” is executed, or until the “DR_ENV” primitive is executed.

**See also:**

## ReadTIM

Produces TIM header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**ReadTIM**
**TIM_IMAGE** ***ReadTIM*** (**timing*)
**TIM_IMAGE** **timing*

### Arguments

*timing* TIM_IMAGE AS structure pointer

### Explanation

The ReadTIM() function sets the members of the TIM_IMAGE structure pointed to by *timing* according to the data specified by the most recent OpenTIM() function.

### Return value

Returns the value of *timing* if succesful; returns 0 if it fails.

### Remarks

**See also:**

## ReadTMD

Reads contents of TMD primitives.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**TMD_PRIM *ReadTMD** (**tmdprim*)
**TMD_PRIM** **tmdprim*;

### Arguments

| | |
|---|---|
| *tmdprim* | Pointer to printer for TMD-PRIM structure. |

### Explanation

The ReadTMD() function sets the members of the TMD_PRIM structure pointed to by *tmdprim* according to the data specified by the most recent OpenTMD() function.

### Return value

Returns *tmdprim* if successful; 0 if fails.

### Remarks

Note that the TMD_PRIM structure includes fields that are not used for all types of objects. ReadTIM() copies only those fields that are valid for the current object.

### See also:

## ResetGraph

Initializes drawing engine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**int ResetGraph** (*mode*)
**int** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Reset mode |

### Explanation

This function resets the graphic system in mode specified by mode. Possible setting of more are listed below.

**Table 6–2**

| Mode | Operation |
|---|---|
| 0 | Complete reset. The drawing environment and display environment are initialized. |
| 1 | Cancels the current drawing and flushes the command buffer. |
| 3 | Initializes the drawing engine while preserving the current display environment (i.e. the screen is not cleared or the screen mode changed). |

### Return value

None.

### Remarks

**See also:**

## SetDefDispEnv

Sets display environment structure members and screen display area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**DISPENV*SetDispEnv** (**disp*, *x*, *y*, *w*, *h*)
**DISPENV** **disp*;
**int** *x*, *y*;
**int** *w*, *h*;

### Arguments

| | |
|---|---|
| *disp* | Pointer to display environment |
| *x*, *y* | Upper left corner of display area |
| *w*, *h* | Width and height of the display area |

### Explanation

This function sets the members of a DISPENV (display environment) structure. The new display area is specified using the coordinates within the frame buffer of the top left corner, along with the width and height, of the desired rectangle.

**Table 6–3**

| Member | Content | Value |
|---|---|---|
| disp | Display area | (x, y, w, h) |
| screen | Screen display area | (0, 0)-(256, 240) |
| ininter | Interlace flag | 0 |
| isrgb24 | 24-bit mode flag | 0 |

### Return value

The return value is the starting pointer of the display environment which has been set.

### Remarks

This function does not actually change the display environment. It merely sets the members of the specified structure as desired. Use the PutDispEnv() function with this structure to change the actual environment.

**See also:**

## SetDefDrawEnv

Set standard drawing environment structure.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Syntax**

**DRAWENV*SetDefDrawEnv** (*env*, *x*, *y*, *w*, *h*)
**DRAWENV** **env*;
**int** *x*, *y*, *w*, *h*;

**Arguments**

| | |
|---|---|
| *env* | Pointer to drawing environment |
| *x*, *y* | Upper left corner of drawing area |
| *w*, *h* | Width and height of drawing area |

**Explanation**

This function sets the drawing area members of a DRAWENV (drawing environment) structure. The new drawing area is specified using the coordinates within the frame buffer of the top left corner, along with the width and height, of the desired rectangle.

Table 6–4

| Member | Content | Value |
|---|---|---|
| clip | Drawing area | (x, y, w, h) |
| ofs[2] | Drawing offset | (x, y) |
| tw | Texture window | (0, 0, 0, 0) |
| tpage | Texture page (tp, abr, tx, ty) | (0, 0, 640, 0) |
| dtd | Dither processing flag | 1 (ON) |
| dfe | Permission flag for drawing | 1 (drawing on display area is inhibited) |
| isbg | Draw area clear flag | 0 (clear: OFF) |
| r0, g0, b0 | Background color | (0, 0, 0) |

**Return value**

The return value is the starting pointer of the drawing environment which has been set.

**Remarks**

This function does not actually change the drawing environment. It merely sets the members of the specified structure as desired. Use the PutDrawEnv() function with this structure to change the actual environment.

**See also:**

## SetDispMask

Sets and cancels display mask.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Syntax**

**void SetDispMask** (*mask*)
**int** *mask*;

**Arguments**

| | |
|---|---|
| *mask* | Display mask |

### Explanation

This function puts display mask into the status specified by *mask*. Any of the following can be designated as *mask*:

**Table 6–5**

| Mask | Operation |
|---|---|
| 0 | Not displayed on screen |
| 1 | Displayed on screen |

### Return value

None.

### Remarks

**See also:**

## SetDrawArea

Initializes the content of drawing area setting primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetDrawArea** (**p*, **r*)
**DR_AREA** **p*;
**RECT** **r*;

### Arguments

| | |
|---|---|
| *p* | Pointer to drawing area setting primitive |
| *r* | Pointer to drawing area |

### Explanation

Initializes a DR_AREA primitive. By using AddPrim() to insert a DR_AREA primitive into your primitive list, it is possible to change part of your drawing environment in the middle of drawing.

### Return value

None.

### Remarks

**See also:**

## SetDrawEnv

Initializes the content of the drawing environment change primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetDrawEnv** (**dr_env*, **env*)
**DR_ENV** **dr_env*;
**DRAWENV** **env*;

### Arguments

| | |
|---|---|
| *dr_env* | Pointer to drawing environment change primitive. |
| *env* | Pointer to drawing environment structure in which the drawing environment is described. |

### Explanation

Initializes a DR_ENV primitive using the values contained in a DRAWENV structure. By using AddPrim() to insert a DR_ENV primitive into your primitive list, it is possible to change part of your drawing environment in the middle of drawing.

### Return value

None.

### Remarks

The DR_ENV primitive uses the same information as the DRAWENV structure, but the data format is different and the DRAWENV structure cannot be used as a primitive. When the DR_ENV primitive is executed, the previous drawing environment settings are destroyed.

**See also:**

## SetDrawMode

Initializes the content of a drawing mode primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetDrawMode** (**p*, *dfe*, *dtd*, *tpage*, **tw*)
**DR_MODE** **p*;
**int** *dfe*, *dtd*, *tpage*;
**RECT** **tw*;

### Arguments

| | |
|---|---|
| *p* | Pointer to drawing mode primitive |
| *dfe* | Dither processing flag: 0: OFF, 1: ON |
| *dtd* | Flag for drawing to a display area 0: OFF, 1: ON |
| *tpage* | Texture page |
| *tw* | Pointer to texture window |

### Explanation

Initializes a DR_MODE primitive using the specified values. By using AddPrim() to insert a DR_MODE primitive into your primitive list, it is possible to change part of your drawing environment in the middle of drawing.

If *tw* is 0, the texture window is not changed.

See the table below for allowable values for the *dtd* and *dfe* parameters.

**Table 6–6**

| *dtd* | Action |
|---|---|
| 0 | Dither processing not performance |
| 1 | Dither processing performance |

| *dfe* | Action |
|---|---|
| 0 | No drawing in display area |
| 1 | Drawing in display area |

**Return value**

None.

**Remarks**

**See also:**

## SetDrawMove

Initializes the contents of a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Syntax**

**void SetDrawMove (**
**DR_MOVE** **p*
**)**

**Arguments**

*p* Pointer to primitive that sets the rectangular area copy offset

**Explanation**

Initializes the rectangular area copy primitive. Due to the initialized primitive being registered to OT by AddPrim (), it can perform the same processing (copy of rectangular area) as MoveImage.

**Return value**

None.

**Remarks**

**See also:**

## SetDrawOffset

Initializes the content of drawing offset setting primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

**Syntax**

**void SetDrawOffset** (**p*, **ofs*)

**DR_OFFSET** **p*;
**u_short** **ofs*;

### Arguments

| | |
|---|---|
| *p* | Pointer to drawing offset setting primitive |
| *ofs* | Pointer to drawing offset |

### Explanation

Initializes a DR_OFFSET primitive using the specified values. By using AddPrim() to insert a DR_OFFSET primitive into your primitive list, it is possible to change part of your drawing environment in the middle of drawing.

### Return value

None.

### Remarks

**See also:**

## SetDrawTPage

Initializes the contents of texture page change primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetDrawTPage (DR_TPAGE** **p*, **int** *dfe*, **int** *dtd*, **int** *tpage***)**

### Arguments

| | |
|---|---|
| *p* | Pointer to texture page change primitive |
| *dtd* | Dither processing flag:<br>0: dither processing not performed<br>1: dither processing performed |
| *dfe* | Flag for drawing to a display area<br>0: no drawing in display area<br>1: drawing in display area |
| *tpage* | Texture page |

### Explanation

Initializes a DR_TPAGE primitive using the specified values. By using AddPrim() to insert a DR_TPAGE primitive into your primitive list, it is possible to change the current texture page in the middle of drawing. This is useful for controlling the textures of certain primitives that do not contain a texture page field.

### Return value

None.

### Remarks

**See also:**

## SetDumpFnt

Defines stream for onscreen dump.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetDumpFnt** (*id*)
**long** *id*;

### Arguments

| | |
|---|---|
| *id* | Print stream ID |

### Explanation

This function sets the print stream for debug printing. The output of the debug printing functions can then be carried out in relation to the stream specified in *id*.

### Return value

None.

### Remarks

The actual display is executed by the FntFlush() function.

**See also:**

## SetGraphDebug

Sets debugging level.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetGraphDebug** (*level*)
**int** *level*;

### Arguments

| | |
|---|---|
| *level* | Debugging level |

### Explanation

Set a debugging level for the graphics system. Any of the following can be designated as *level*:

**Table 6–7**

| Level | Operation |
|---|---|
| 0 | No checks are performed. (Highest speed mode) |
| 1 | Checks coordinating registered and drawn primitives. |
| 2 | Registered and drawn primitives are dumped. |

### Return value

The previously set debug level.

### Remarks

See also:

## SetLineF2, SetLineF3, SetLineF4

Initialize Line_F2 primitive/ Initialize Line_F3 primitive/ Initialize Line_F4 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetLineF2** (**p*)
**LINE_F2** **p*;
**void SetLineF3** (**p*)
**LINE_F3** **p*;
**void SetLineF4**
**LINE_F4** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |

### Explanation

These functions initialize the primitives specified by *p*.

### Return value

None.

### Remarks

See also:

## SetLineG2, SetLineG3, SetLineG4

Initialize Line_G2 primitive/ Initialize Line_G3 primitive/ Initialize Line_G4 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetLineG2** (**p*)
**LINE_G2** **p*;
**void SetLineG3** (**p*)
**LINE_G3** **p*;
**void SetLineG4** (**p*)
**LINE_G4** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |

### Explanation

These functions initialize the primitives specified by *p*.

### Return value

None.

### Remarks

**See also:**

## SetPolyF3, SetPolyF4

Initialize Poly_F3 primitive/ Initialize Poly_F4 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetPolyF3** (**p*)
**POLY_F3** **p*;
**void SetPolyF4** (**p*)
**POLY_F4** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |

### Explanation

These functions initialize the primitive specified by *p*.

### Return value

None.

### Remarks

**See also:**

## SetPolyFT3, SetPolyFT4

Initialize Poly_FT3 primitive/ Initialize Poly_FT4 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetPolyFT3** (**p*)
**POLY_FT3** **p*;
**void SetPolyFT4** (**p*)
**POLY_FT4** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |

### Explanation

These functions initialize the primitive specified by *p*.

### Return value

None.

### Remarks

**See also:**

## SetPolyG3, SetPolyG4

Initialize Poly_G3 primitive/ Initialize Poly_G4 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetPolyG3** (**p*)
**POLY_G3** **p*;
**void SetPolyG4** (**p*)
**POLY_G4** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |

### Explanation

These functions initialize the primitive specified by *p*.

### Return value

None.

### Remarks

**See also:**

## SetPolyGT3, SetPolyGT4

Initialize Poly_GT3 primitive/ Initialize Poly_GT4 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetPolyGT3** (**p*)
**POLY_GT3** **p*;
**void SetPolyGT4** (**p*)

**POLY_GT4** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |

### Explanation

These functions initialize the primitive specified by *p*.

### Return value

None.

### Remarks

**See also:**

## setRECT

Set rectangular area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setRECT** (*r*, *x*, *y*, *w*, *h*)

### Arguments

| | |
|---|---|
| *r* | Pointer to RECT structure |
| *x*, *y* | Upper left point of rectangular area |
| *w*, *h* | Size of rectangular area |

### Explanation

Sets the *x*, *y*, *w*, and *h* values of the RECT structure *r*.

### Return value

None.

### Remarks

**See also:**

## setRGB0, setRGB1, setRGB2, setRGB3

Initialize *r0*, *g0*, and *b0* fields of a primitive/ Initialize *r1*, *g1*, and *b1* fields of a primitive/ Initialize *r2*, *g2*, and *b2* fields of a primitive/ Initialize *r3*, *g3*, and *b3* fields of a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setRGB0** (*p*, *r0*, *g0*, *b0*)

**setRGB1** (*p*, *r1*, *g1*, *b1*)
**setRGB2** (*p*, *r2*, *g2*, *b2*)
**setRGB3** (*p*, *r3*, *g3*, *b3*)

### Arguments

| | |
|---|---|
| *p* | Primitive pointer |
| *r*, *g*, *b* | RGB members of primitive. |

### Explanation

These macros set the values for the RGB members of the primitive *p*.

### Return value

None.

### Remarks

These are macros, so there is no dependence on the primitive type.

**See also:**

## SetSemiTrans

Sets the semi-transparent attribute of a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetSemiTrans** (**p*, *abe*)
**unsigned long** **p*;
**long** *abe*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |
| *abe* | Semi-transparent flag<br>0: semitransparent OFF<br>1: Semitransparent ON |

### Explanation

This function sets the semi-transparent attribute of the primitive specified by *p* to the value specified by the *abe* parameter. If semi-transparent mode is enabled, then semi-transparent pixels are drawn as specified by the table below.

**Table 6–8**

| Primitive | Pixels subjected to semitransparent processing |
|---|---|
| POLY_FT3/POLY_FT4 | Pixels for which the topmost bit of the corresponding texture pixel is “1” |
| POLY_GT3/POLY_GT4 | Pixels for which the topmost bit of the corresponding texture pixel is “1” |
| SPRT/SPRT_8/SPRT_16 | Pixels for which the topmost bit of the corresponding texture pixel is “1” |
| Other drawing primitives | All Pixels |

### Return value

None.

### Remarks

Semi-transparent pixels are calculated from the foreground pixels Pf and background pixels Pb as follows:

P = F x Pf + B x Pb

The rate (F, B) of semi-transparency is designated by the member *tpage* in the primitive. Drawing speed is reduced because semi-transparency requires reading of background brightness values. Therefore, do not draw primitives with semi-transparent mode turned on unless they are to be displayed that way.

**See also:**

## SetShadeTex

Inhibiting shading function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetShadeTex** (**p*, *tge*)
**unsigned long** **p*;
**long** *tge*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |
| *tge* | Unshaded flag<br>0: Shading is performed<br>1: Shading is not performed |

### Explanation

This function sets the shading attribute of the primitive pointed to by *p* to the value specified by the *tge* parameter.

When texture and shading are both ON, each pixel in the polygon is calculated as shown below from the pixel value “T” of the corresponding texture pattern, and the brightness value “L” correspondong to the pixel value “T”.

P = (T•L)/128

When “L” = 128, the brightness value of the texture pattern is drawn as it is. If the value results in an overflow, the pixel value is clipped to 255.

When *tge* = 1, the brightness value is not divided, and the texture pattern value is used, as it is, as the pixel value.

### Return value

None.

### Remarks

This function cannot be used for primitives other than “POLY_FT3”, “POLY_FT4”, “SPRT”, “SPRT_8”, and “SPRT_16”.

**See also:**

## SetSprt, SetSprt8, SetSprt16

Initialize a SPRT primitive/ Initialize a SPRT8 primitive/ Initialize a SPRT16 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetSprt8** (**p*)
**SPRT_8** **p*;
**void SetSprt16** (**p*)
**SPRT_16** **p*;
**void SetSprt** (**p*)
**SPRT** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address |

### Explanation

These functions initialize the primitives specified by *p*. Details are given below.

Table 6–9

| Function name | Sprite size | Primitive |
|---|---|---|
| SetSprt8 | 8 x 8 | SPRT_8 |
| SetSprt16 | 16 x 16 | SPRT_16 |
| SetSprt | Can be set at will using values of members h, w. (0 < h , 255, 0 < w < 255) | SPRT |

### Return value

None.

### Remarks

The SPRT... primitives are faster than POLY_FT4. TILE is also faster than POLY_F4.

**See also:**

## SetTexWindow

Initializes the content of a texture window primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetTexWindow** (**p*, **tw*)
**DR_TWIN** **p*
**RECT** **tw*

### Arguments

| | |
|---|---|
| *p* | Pointer to texture window primitive |
| *tw* | Pointer to texture window |

### Explanation

Initializes a DR_TWIN primitive using the specified values. By using AddPrim() to insert a DR_TWIN primitive into your primitive list, it is possible to change the current texture window in the middle of drawing.

### Return value

None.

### Remarks

**See also:**

## SetTile, SetTile1, SetTile8, SetTile16

Initialize a TILE primitive/ Initialize a TILE1 primitive/ Initialize a TILE8 primitive/ Initialize a TILE16 primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void SetTile** (**p*)
**TILE** **p*;
**void SetTile** (**p*)
**TILE_1** **p*;
**void SetTile** (**p*)
**TILE_8** **p*;
**void SetTile** (**p*)
**TILE_16** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to primitive start address. |

### Explanation

These functions initialize the primitives specified by *p*. Details are given below.

**Table 6–10**

| Function name | Tile size | Primitive size |
|---|---|---|
| SetTile1 | 1 x 1 | TILE |
| SetTile8 | 8 x 8 | TILE_1 |
| SetTile16 | 8 x 8 | TILE_8 |
| SetTile | Can be set at will using values of members h, w. (0 < h , 255, 0 < w < 255) | TILE_16 |

### Return value

None.

### Remarks

The SPRT... primitives are faster than POLY_FT4. TILE is also faster than POLY_F4.

**See also:**

## setUV0, setUV3, setUV4

Set the *u0* and *v0* parameters of a primitive/ Set the *u3* and *v3* parameters of a primitive/ Set the *u4* and *v4* parameters of a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setUV0** (**p*, *u0*, *v0*)
**setUV3** (**p*, *u0*, *v0*, *u1*, *v1*, *u2*, *v2*)
**setUV4** (**p*, *u0*, *v0*, *u1*, *v1*, *u2*, *v2*, *u3*, *v3*)

### Arguments

| | |
|---|---|
| *p* | Primitive pointer. |
| *u*, *v* | UV members of primitive. |

### Explanation

These macros set the values of the appropriate UV fields of the primitive *p*.

### Return value

None.

### Remarks

These are C preprocessor macros and can be used with any primitive or structure with the appropriate fields.

**See also:**

## setUVWH

Sets the UV members of a primitive structure.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setUVWH** (**p*, *u0*, *v0*, *w*, *h*)

### Arguments

| | |
|---|---|
| *p* | Primitive pointer. |
| *u0*, *v0* | Upper left corner of primitive texture |
| *w*, *h* | Width and height of primitive texture. |

### Explanation

This macro sets the *u0*, *v0*, *u1*, *v1*, *u2*, *v2*, *u3*, and *v3* fields of a primitive structure to represent the corners of the rectangle specified by the input parameters.

### Return value

None.

### Remarks

This is a C preprocessor macro and can be used with any primitive or structure with the appropriate fields.

**See also:**

## setVector

Setting a vector value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setVector** (**v*, *x*, *y*, *z*)

### Arguments

| | |
|---|---|
| *v* | Pointer to a vector |
| *x*, *y*, *z* | Coordinate values |

### Explanation

Sets the (x, y, z) value for VECTOR/SVECTOR.

### Return value

None.

### Remarks

setVector() is not dependent on vector format because it is a macro instruction.

Operation differs between:

a) setVector ( (SVECTOR*)v, x, y, z)
b) setVector ( (VECTOR *)v, x, y, z)

**See also:**

## SetVideoMode

Declares current video signaling system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *3.1* | *7/31/96* |

### Syntax

**long SetVideoMode** (**long** *mode*)

### Arguments

| | |
|---|---|
| *mode* | Video signaling system mode |

### Explanation

Declares the video signaling system indicated by mode to the libraries.

**Table 6–11**

| Mode | Contents |
|---|---|
| MODE_NTSC | NTSC system video signaling system |
| MODE_PAL | PAL system video signaling system |

Related libraries will be able to conform to the actions of the declared video signaling system environment.

### Return value

Previously-set video signaling system mode.

### Remarks

Gets called in advance of all library functions.

**See also:**

## setVWH

Sets the UV members of the 4-point designated primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setVWH** (**p*, *u0*, *v0*, *w*, *h*)

### Arguments

| | |
|---|---|
| *p* | Primitive pointer. |
| *u0*, *v0* | Left top point of primitive texture |
| *w*, *h* | Width and height of primitive texture |

### Explanation

Designates the (u0, v0) - (u0 + w, v0 + h) on the diagonal line containing each coordinate of the rectangle as the (u0, v0). . (u3, v3) members of the primitive.

### Return value

None.

### Remarks

setVWH is a macro, so there is no dependence on the primitive model.

Cannot be used in the sprite primitive.

**See also:**

## setXY0, setXY2, setXY3, setXY4

Set the *x0* and *y0* parameters of a primitive/ Set the *x2* and *y2* parameters of a primitive/ Set the *x3* and *y3* parameters of a primitive/ Set the *x4* and *y4* parameters of a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setXY0** (**p*, *x0*, *y0*)
**setXY2** (**p*, *x0*, *y0*, *x1*, *y1*)
**setXY3** (**p*, *x0*, *y0*, *x1*, *y1*, *x2*, *y2*)
**setXY4** (**p*, *x0*, *y0*, *x1*, *y1*, *x2*, *y2*, *x3*, *y3*)

### Arguments

| | |
|---|---|
| *p* | Primitive pointer |
| *x*, *y* | XY members of primitive |

### Explanation

These macros set the values for the XY members of the primitive.

### Return value

None.

### Remarks

These are macros, so there is no dependence on the primitive type.

### Remarks

**See also:**

## setXYWH

Sets the XY members of a primitive.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**setXYWH** (**p*, *x0*, *y0*, *w*, *h*)

### Arguments

| | |
|---|---|
| *p* | Primitive pointer. |
| *x0*, *y0* | Upper left corner of primitive. |
| *w*, *h* | Width and height of primitive. |

### Explanation

This macro sets the *x0*, *y0*, *x1*, *y1*, *x2*, *y2*, *x3*, and *y3* fields of a primitive structure to represent the corners of the rectangle specified by the input parameters.

### Return value

None.

### Remarks

This is a C preprocessor macro and can be used with any primitive or structure with the appropriate fields.

**See also:**

## StoreImage

Transfers image data from the frame buffer to main memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**int StoreImage** (**recp*, **p*)
**RECT** **recp*;
**unsigned long** **p*;

### Arguments

| | |
|---|---|
| *recp* | Pointer to destination rectangular area |
| *p* | Pointer to main memory address of destination of transmission |

### Explanation

This function transfers the rectangular portion specified by *recp* from the frame buffer to the address in main memory specified by the *p* parameter.

### Return value

Number in the queue.

### Remarks

Because StoreImage() is a non-blocking function, use the DrawSync() function to determine when the operation has completed.

The transfer areas at the source and destination are not affected by the drawing environment (clip, offset). The source area must be located within a drawable area (0, 0) - (1023, 511).\

**See also:**

## TermPrim

Terminates a primitive list

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgpu.lib* | *Libgpu.h* | *2.x* | *7/31/96* |

### Syntax

**void TermPrim** (**p*)

**unsigned long** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to start address of a primitive |

### Explanation

This function sets the tag pointer of the primitive specified by *p* to point at a special terminator value that will signal the end of the list when it is executed. Any primitives already pointed to by *p* will be removed from the list.

### Return value

None.

### Remarks

**See also:**

## VSync

Waits for the next vertical blank, or returns the vertical blank counter value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *2.x* | *7/31/96* |

### Syntax

**int VSync**(*mode*)
**int** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Mode |

### Explanation

This function waits for vertical blank using the method specified by the *mode* parameter, as defined below.

**Table 6–12**

| Mode | Operation |
|---|---|
| 0 | Blocks until vertical sync is generated |
| 1 | Returns time elapsed from the point in time VSync() is last called in horizontal sync units |
| n (n>1) | Blocks from the point in time VSync() is last called until *n* number of vertical syncs are generated. |
| -n (n>0) | Returns absolute time after program boot in vertical sync interval units. |

### Return value

Mode value is as listed below.

**Table 6–13**

| Mode | Return value |
|---|---|
| mode>=0 | Time elapsed from the point in time that Vsync() is last called (horizontal blanking units) |
| mode<0 | Time elapsed after program boot (vertical blanking units) |

### Remarks

The Vsync() function may generate a timeout if long blocking periods are specified. To prevent deadlocks, rather than using Vsync() to block for an especially long time (say more than 4 vertical blank periods), have your program poll VSync(-1) in a loop instead.

### See also:

## VSyncCallback

Defines a function to be executed during each vertical blank period.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *2.x* | *7/31/96* |

### Syntax

**void VSyncCallback** (**func*)
**void** (**func)()*;

### Arguments

| | |
|---|---|
| *func* | Pointer to callback function |

### Explanation

Specifies that the routine at address *func* should be executed at the start of the vertical blank interrupt. If *func* is 0, then any previous callback routine is disabled.

### Return value

None.

### Remarks

Subsequent interrupts will be masked inside *func*. Therefore, it is necessary to return quickly after performing necessary processes using *func*.

Although the specified function is called during an interrupt, it is not the actual interrupt handler. It should be written as a normal subroutine that will be called by the main interrupt handler.

### See also:

# Chapter 7: Basic Geometry Library
## Table of Contents

## CRVECTOR3

Triangular recursive vector data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **RVECTOR** *r01*, *r12*, *r20*;
    **RVECTOR** **r0*, **r1*, **r2*;
    **unsigned long** **rtn*;
**} CRVECTOR3;**

### Members

| | |
|---|---|
| *r01*, *r12*, *r20* | Division vertex vector data |
| *r0*, *r1*, *r2* | Pointer to division vector data |
| *rtn* | Pointer to return address for assembler |

### Explanation

### Remarks

**See also:**

## CRVECTOR4

Quadrilateral recursive vector data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **RVECTOR** *r01*, *r02*, *r31*, *r32*, *rc*;
    **RVECTOR** **r0*, **r1*, **r2*, **r3*;
    **unsigned long** **rtn*;
**} CRVECTOR4;**

### Members

| | |
|---|---|
| *r01*, *r02*, *r31*, *r32*, *rc* | Division vertex vector data |
| *r0*, *r1*, *r2*, *r3* | Pointer to division vertex vector data |
| *rtn* | Pointer to return address for assembler |

### Explanation

### Remarks

**See also:**

## CVECTOR

Character vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

**typedef struct {**
    **unsigned char** *r*, *g*, *b*, *cd*;
**}**;

### Members

| | |
|---|---|
| *r*, *g*, *b* | Color palette |
| *cd* | GPU code |

### Explanation

### Remarks

**See also:**

## DIVPOLYGON3

Triangular division buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
  **unsigned long** *ndiv*;
  **unsigned long** *pih*, *piv*;
  **unsigned short** *clut*, *tpage*;
  **CVECTOR** *rgbc*;
  **unsigned long** **ot*;
  **RVECTOR** *r0*, *r1*, *r2*;
  **CRVECTOR3** *cr*[5];
**} DIVPOLYGON3;**

### Members

| | |
|---|---|
| *ndiv* | Number of divisions |
| pih, piv | Clip area specification (display screen resolution) |
| *clut* | CLUT |
| *tpage* | Texture page |
| *rgbc* | Code + RGB color |
| *ot* | Pointer to OT |
| *r0*, *r1*, *r2* | Division vertex vector data |
| *cr* | Triangular recursive vector data |

### Explanation

### Remarks

**See also:**

## DIVPOLYGON4

Quadrilateral recursive vector data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **unsigned long** *ndiv*;
    **unsigned long** *pih*, *piv*;
    **unsigned short** *clut*, *tpage*;
    **CVECTOR** *rgbc*;
    **unsigned long** **ot*;
    **RVECTOR** *r0*, *r1*, *r2*, *r3*;
    **CRVECTOR4** *cr*[5];
**} DIVPOLYGON4;**

### Members

| | |
|---|---|
| *ndiv* | Number of divisions |
| *pih*, *piv* | Clip area specification (display screen's resolution) |
| *clut* | CLUT |
| *tpage* | Texture page |
| *rgbc* | Code + RGB color |
| *ot* | Pointer to OT |
| *r0*, *r1*, *r2*, *r3* | Division vertex vector data |
| *cr* | Quadrilateral recursive vector data |

### Explanation

### Remarks

**See also:**

## DVECTOR

2D vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **short** *vx*, *vy*;
**} DVECTOR;**

### Members

| | |
|---|---|
| *vx*, *vy* | Vector coordinates |

### Explanation

### Remarks

**See also:**

## EVECTOR

Clip vector data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **SVECTOR** *v*;
    **VECTOR** *sxyz*;
    **DVECTOR** *sxy*;
    **CVECTOR** *rgb*;
    **short** *txuv*, *pad*;
    **long** *chx*, *chy*;
**} EVECTOR;**

### Members

| | |
|---|---|
| *v* | Local object 3D vertex |
| *sxyz* | Screen 3D vertex |
| *sxy* | Screen 2D vertex |
| *rgb* | Color palette |
| *txuv*, *pad* | Texture mapping data |
| *chx*, *chy* | Clip area data |

### Explanation

### Remarks

**See also:**

## MATRIX

Matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

**struct MATRIX {**
  **short** *m* [3][3];
  **long** *t* [3];
**}**;

### Members

| | |
|---|---|
| *m* | 3 x 3 matrix coefficient value |
| *t* | Parallel transfer volume |

### Explanation

Specifies each component on the MATRIX *m*[i][j]. Specifies the transfer volume after conversion on the MATRIX *t* [l]. Pay attention to the differing word lengths on *m* and *t.*

The GTE essentially performs the following multiply and accumulate calculations from the MATRIX structure.

a) RotTrans system function (function group which does not perform coordinate conversion). Performs only basic matrix calculations and vector addition.

MATRIXm

SVECTORxi

SVECTORxo

$$\begin{bmatrix} xo.vx \\ xo.vy \\ xo.vz \end{bmatrix} = \begin{bmatrix} m.m[0][0] & m.m[0][1] & m.m[0][2] \\ m.m[1][0] & m.m[1][1] & m.m[1][2] \\ m.m[2][0] & m.m[2][1] & m.m[2][2] \end{bmatrix} \begin{bmatrix} xi.vx \\ xi.vy \\ xi.vz \end{bmatrix} + \begin{bmatrix} m.t[0] \\ m.t[1] \\ m.t[2] \end{bmatrix}$$

b) RotTransPers system function (function group which performs coordinate conversion). In addition to the (a) calculation, perspective conversion (division by z) is performed at the same time.

MATRIXm

SVECTORxi

SVECTORxo

SVECTOR x2

long h

$$\begin{bmatrix} xo.vx \\ xo.vy \\ xo.vz \end{bmatrix} = \begin{bmatrix} m.m[0][0] & m.m[0][1] & m.m[0][2] \\ m.m[1][0] & m.m[1][1] & m.m[1][2] \\ m.m[2][0] & m.m[2][1] & m.m[2][2] \end{bmatrix} \begin{bmatrix} xi.vx \\ xi.vy \\ xi.vz \end{bmatrix} + \begin{bmatrix} m.t[0] \\ m.t[1] \\ m.t[2] \end{bmatrix}$$

x2.vx = (h*xo.vx) / xo.vz

x2.vy = (h*yo.vy) / xo.vz

### Remarks

**See also:**

## POL3

Triangle polygon.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

**struct POL3 {**
    **short** *sxy* [3][2];
    **short** *sz* [3][2];
    **short** *uv* [3][2];
    **short** *rgb* [3][3];
    **short** *code*;
**}**;

### Members

| | |
|---|---|
| *sxy* | Screen coordinates |
| *sz* | Screen coordinates |
| *uv* | Texture coordinates |
| *rgb* | RGB value |
| *code* | Code |

**Table 7–1**

| Code | Values |
|---|---|
| F3 | 1 |
| TF3 | 2 |
| G3 | 3 |
| TG3 | 4 |

### Explanation

### Remarks

**See also:**

## POL4

Four-sided polygon.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

```
struct POL4 {
    short sxy [4][2];
    short sz [4][2];
    short uv [4][2];
    short rgb [4][3];
    short code;
};
```

### Members

| | |
|---|---|
| *sxy* | Screen coordinates |
| *sz* | Screen coordinates |
| *uv* | Texture coordinates |
| *rgb* | RGB value |
| *code* | Code |

**Table 7–2**

| Code | Values |
|---|---|
| F4 | 5 |
| TF4 | 6 |
| G4 | 7 |
| TG4 | 8 |

### Explanation

### Remarks

**See also:**

## RVECTOR

Division vertex vector data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **SVECTOR** *v*;
    **unsigned char** *uv* [2];
    **unsigned short** *pad*;
    **CVECTOR** *c*;
    **DVECTOR** *sxy*;
    **unsigned long** *sz*;
**} RVECTOR;**

### Members

| | |
|---|---|
| *v* | Local object 3D vertex |
| *uv* | Texture mapping data |
| *c* | Vertex color palette |
| *sxy* | Screen 2D vertex |
| *sz* | Clip Z-data |

### Explanation

### Remarks

**See also:**

## SPOL

Vertex information.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

**struct SPOL {**
    **short** *xy* [3];
    **short** *uv* [2];
    **short** *rgb* [3];
**}**;

### Members

| | |
|---|---|
| *xy* | XY coordinates |
| *uv* | UV coordinates |
| *rgb* | RGB value |

### Explanation

### Remarks

**See also:**

## SVECTOR

Short vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

```
struct SVECTOR {
    short vx, vy;
    short vz, pad;
};
```

### Members

| | |
|---|---|
| *vx*, *vy*, *vz* | Vector coordinates |
| *pad* | System reserved |

### Explanation

### Remarks

**See also:**

## TMESH

Triangle mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

**struct TMESH {**
    **SVECTOR** **v*;
    **SVECTOR** **n*;
    **SVECTOR** **u*;
    **CVECTOR** **c*;
    **unsigned long** *len*;
**};**

### Members

| | |
|---|---|
| *v* | Pointer to vertex string |
| *n* | Pointer to normal string |
| *u* | Pointer to texture string |
| *c* | Pointer to RGB string |
| *len* | Mesh length |

### Explanation

### Remarks

**See also:**

## VECTOR

Vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Structure

**struct VECTOR {**
    **long** *vx*, *vy*, *vz*, *pad*;
**}**;

### Members

|  |  |
|---|---|
| *vx*, *vy*, *vz* | Vector coordinates |
| *pad* | System reserved |

### Explanation

### Remarks

**See also:**

## ApplyMatrix

Multiply a vector by a matrix. The vector is in effect rotated and then translated.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**VECTOR*** **ApplyMatrix** (**m*, **v0*, **v1*)
**MATRIX** **m*;
**SVECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix to be multiplied (input) |
| *v0* | Pointer to short vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function multiplies the matrix *m* by the short vector *v0* beginning with the rightmost end. The result is saved in the vector *v1*.

The argument format is as follows:

| | |
|---|---|
| m -> m [i][j] | : (1, 3, 12) |
| v0 -> vx, vy, vz: | :(1, 15, 0) |
| v1 -> vx, vy, vz: | :(1, 31, 0) |

### Return value

This function returns *v1*.

### Remarks

The function destroys the constant rotation matrix.

**See also:**

## ApplyMatrixLV

Multiply a vector by a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**VECTOR*** **ApplyMatrixLV** (**m*, **v0*, **v1*)
**MATRIX** **m*;
**VECTOR** **v0*, **v1*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix to be multiplied (input) |
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function destroys the rotation matrix.

This function multiplies matrix m by vector *v0* beginning from the rightmost end. The result is saved in vector *v1*.

| | |
|---|---|
| m -> m [i][j] | : (1, 3, 12) |
| v0 -> vx, vy, vz | : (1, 31, 0) |
| v1 -> vx, vy, vz | : (1, 31, 0) |

### Return value

*v1*

### Remarks

This function destroys the rotation matrix.

ApplyMatrixLV is a 16 x 32 bit multiplier which uses the GTE.

**See also:**

## ApplyMatrixSV

Multiply a vector by a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**SVECTOR*** **ApplyMatrix** (**m*, **v0*, **v1*)
**MATRIX** **m*;
**SVECTOR** **v0*, **v1*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix to be multiplied (input) |
| *v0* | Pointer to short vector (input) |
| *v1* | Pointer to short vector (output) |

### Explanation

This function multiplies matrix *m* by short vector *v0* beginning at the rightmost end. The result is saved in the short vector *v1*.

| | |
|---|---|
| m -> m [i][j] | : (1, 3, 12) |
| v0 -> vx, vy, vz | : (1, 15, 0) |
| v1 -> vx, vy, vz | : (1, 15, 0) |

### Return value

*v1*

### Remarks

This function destroys the rotation matrix.

**See also:**

## ApplyRotMatrix

Multiply a vector by a constant rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**VECTOR* ApplyRotMatrix** (**v0*, **v1*)
**SVECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to short vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function multiplies a constant rotation matrix by short vector *v0* beginning at the rightmost end. The result is saved in vector *v1*.

v0 -> vx, vy, vz : (1, 15, 0)

v1 -> vx, vy, vz : (1, 31, 0)

### Return value

*v1*

### Remarks

**See also:**

## ApplyRotMatrixLV

Multiplies a vector by a constant rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *10/23/96* |

### Syntax

**VECTOR*** **ApplyRotMatrix** (**v0*, **v1*)
**VECTOR** **v0;* */* Input: Vector */*
**VECTOR** **v1;* */* Output: Vector */*

### Arguments

| | |
|---|---|
| *v0* | Pointer to long vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function multiplies a constant rotation matrix by long vector *v0* beginning at the rightmost end. The result is saved in vector *v1*.

v0 -> vx, vy, vz : (1, 31, 0)

v1 -> vx, vy, vz : (1, 31, 0)

### Return value

*v1*

### Remarks

**See also:**

## ApplyTransposeMatrixLV

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**VECTOR* ApplyTransposeMatrixLV** (**m*, **v0*, **v1*)
**MATRIX** **m*;
**VECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix to be multiplied |
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |

### Explantation

### Return value

### Remarks

**See also:**

## AverageZ3

Average of three values.

### Syntax

**long AverageZ3** (*sz0*, *sz1*, *sz2*)
**long** *sz0*, *sz1*, *sz2*;

### Arguments

| | |
|---|---|
| *sz0*, *sz1*, *sz2* | Input values |

### Explanation

This function calculates an average of three values *sz0*, *sz1*, and *sz2*.

The argument format is as follows:

| | |
|---|---|
| *sz0*, *sz1*, *sz2* | : (0, 16, 0) |
| Return value | : (0, 16, 0) |

### Return value

Average of 1/4 of three values *sz0*, *sz1*, and *sz2.*

### Remarks

**See also:**

## AverageZ4

Average of four values.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long AverageZ4** (*sz0*, *sz1*, *sz2*, *sz3*)
**long** *sz0*, *sz1*, *sz2*, *sz3*;

### Arguments

| | |
|---|---|
| *sz0*, *sz1*, *sz2*, *sz3* | Input values |

### Explanation

This function calculates an average of four values *sz0*, *sz1*, *sz2*, and *sz3*.

The argument format is as follows:

| | |
|---|---|
| *sz0*, *sz1*, *sz2*, *sz3* | : (0, 16, 0) |
| Return value | : (0, 16, 0) |

### Return value

Average of 1/4 of four values *sz0*, *sz1*, *sz2*, *and sz3.*

### Remarks

**See also:**

## catan

Computes the arctangent of angle(a) within 180 degrees.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long catan** (*a*)
**long** *a*;

### Arguments

| | |
|---|---|
| *a* | Value |

### Explanation

This function uses Playstation format (where 4096 = 360 degrees) to find the arctan (between -90 and +90 degrees) of *a*.

The argument format is as follows:

*a*: (1, 19, 12)

Return value: Playstation format (4096 = 360 degrees)

### Return value

atan (a)

### Remarks

**See also:**

## ccos

Computes the cosine of angle a.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long ccos** (*a*)
**long** *a*;

### Arguments

| | |
|---|---|
| *a* | Angle (in Playstation format) |

### Explanation

Find the cosine function of the angle (in Playstation format) (4096 = 360 degrees) using fixed point math (where 4096 = 1.0).

The argument format is as follows:

*a* : Playstation format (4096 = 360 degrees)

Return value : (1, 19, 12)

### Return value

cos (a)

### Remarks

**See also:**

## Clip3F

Three-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip3F** (**v0*, **v1*, **v2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *evmx* | Pointer arrays for clip vector data (20,output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(), and angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices.

### Remarks

**See also:**

## Clip3FP

Three-vertex (triangle) clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip3FP** (**v0*, **v1*, **v2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*;
**EVECTOR** ***ev*mx;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *evmx* | Pointer arrays (for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | FOG effect interpolation value (p) |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip3FT

Three-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip3FT** (**v0*, **v1*, **v2*, **uv0*, **uv1*, **uv2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v1*
**short** **uv0*, **uv1*, **uv2*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2* | Pointer to texture coordinate vector (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> txuv | Texture Mapping Vertex |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip3FTP

Three-vertex clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long** Clip3FTP (**v0*, **v1*, **v2*, **uv0*, **uv1*, **uv2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*;
**short** **uv0*, **uv1*, **uv2*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2* | Pointer to texture coordinate vector (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Object (Local) 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | FOG effect interpolation value (p) |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> txuv | Texture Mapping Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip3G

Three-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip3G** (**v0*, **v1*, **v2*, **rgb0*, **rgb1*, **rgb2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip3GP

Three-vertex clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip3GP** (**v0*, **v1*, **v2*, **rgb0*, **rgb1*, **rgb2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*;
**EVECTOR** ***evmx*; data

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | FOG effect interpolation value (p) |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip3GT

Three-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip3GT** (**v0*, **v1*, **v2*, **uv0*, **uv1*, **uv2*, **rgb0*, **rgb1*, **rgb2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*;
**short** **uv0*, **uv1*, **uv2*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*;
**EVECTOR** ***evmx*; data

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2* | Pointer to texture coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> txuv | Texture Mapping Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip3GTP

Three-vertex clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip3GTP** (**v0*, **v1*, **v2*, **uv0*, **uv1*, **uv2*, **rgb0*, **rgb1*, **rgb2*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*;
**short** **uv0*, **uv1*, **uv2*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*;
**EVECTOR** ***evmx*; data

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2* | Pointer to texture coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a triangle having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | Fog effect interpolation value (p) |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> txuv | Texture Mapping Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4F

Four-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4F** (**v0*, **v1*, **v2*, **v3*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4FP

Four-vertex clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4FP** (**v0*, **v1*, **v2*, **v3*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | FOG effect interpolation value (p) |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4FT

Four-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4FT** (**v0*, **v1*, **v2*, **v3*, **uv0*, **uv1*, **uv2*, **uv3*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**short** **uv0*, **uv1*, **uv2*, **uv3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2*, *uv3* | Pointer to texture coordinate vector (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> txuv | Texture Mapping Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4FTP

Four-vertex clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4FTP** (**v0*, **v1*, **v2*, **v3*, **uv0*, **uv1*, **uv2*, **uv3*, **evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**short** **uv0*, **uv1*, **uv2*, **uv3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2*, *uv3* | Pointer to texture coordinate vector (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | Interpolation value (p) FOG effect |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> txuv | Texture Mapping Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4G

Four-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4G** (**v0*, **v1*, **v2*, **v3*, **rgb0*, **rgb1*, **rgb2*, **rgb3*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*, **rgb3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2*, *rgb3* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4GP

Four-vertex clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4GP** (**v0*, **v1*, **v2*, **v3*, **rgb0*, **rgb1*, **rgb2*, **rgb3*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*, **rgb3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2*, *rgb3* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | interpolation value (p) for FOG effect |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4GT

Four-vertex clipping (without perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4GT** (**v0*, **v1*, **v2*, **v3*, **uv0*, **uv1*, **uv2*, **uv3*, **rgb0*, **rgb1*, **rgb2*, **rgb3*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**short** **uv0*, **uv1*, **uv2*, **uv3*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*, **rgb3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2*, *uv3* | Pointer to texture coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2*, *rgb3* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> txuv | Texture Mapping Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## Clip4GTP

Four-vertex clipping (with perspective transformation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long Clip4GTP** (**v0*, **v1*, **v2*, **v3*, **uv0*, **uv1*, **uv2*, **uv3*, **rgb0*, **rgb1*, **rgb2*, **rgb3*, ***evmx*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**short** **uv0*, **uv1*, **uv2*, **uv3*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*, **rgb3*;
**EVECTOR** ***evmx*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vector (input) |
| *uv0*, *uv1*, *uv2*, *uv3* | Pointer to texture coordinate vector (input) |
| *rgb0*, *rgb1*, *rgb2*, *rgb3* | Pointer to vertex color data (input) |
| *evmx* | Pointer arrays for clip vector data (20, output) |

### Explanation

This function clips six surfaces of a quadrilateral (linked triangle) having vertices *v0*, *v1*, and *v2*, and defined by InitClip(). Angle information is stored in *evmx*. The output number of vertices is returned.

Effective output clip vector data:

| | |
|---|---|
| evmx[i] -> v | Local Object 3D Vertex |
| evmx[i] -> sxyz | Screen 3D Vertex |
| evmx[i] -> sxyz.pad | Fog effect interpolation value (p) |
| evmx[i] -> sxy | Screen 2D Vertex |
| evmx[i] -> rgb | Vertex Color Data |
| evmx[i] -> txuv | Texture Mapping Data |
| evmx[i] -> chx | chx = vz • (hw/2)/h |
| evmx[i] -> chy | chy = vz • (vw/2)/h |

This function reserves the pointer arrays (20 pointer arrays = 80 bytes), including the work area.

### Return value

Output number of vertices

### Remarks

**See also:**

## cln

C logarithm function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long cln** (*a*)
**long** *a*;

### Arguments

*a* Value

### Explanation

This function uses fixed point math (where 4096 = 1.0) to find the fixed point natural logarithm.

Argument format is as follows:

*a* : (1, 19, 12)

Return value : (1, 19, 12)

### Return value

ln (a)

### Remarks

**See also:**

## ColorCol

Finds a local color from a local light vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ColorCol** (**v0*, **v1*, **v2*)
**VECTOR** **v0*;
**CVECTOR** **v1*;
**CVECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to local light vector (input) |
| *v1* | Pointer to primary color vector (input) |
| *v2* | Pointer to color vector (output) |

### Explanation

This function calculates the following:

LC = BK + LCM • v0

v2 = v1 • LC (product of multiplication)

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz : | :(1, 19, 12) |
| *v1* -> r, g, b | : (0, 8, 0) |
| *v2* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## ColorDpq

Finds a local color from a local light vector, and performs depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ColorDpq** (**v0*, **v1*, *p*, **v2*)
**VECTOR** **v0*;
**CVECTOR** **v1*;
**long** *p*;
**CVECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to local light vector (input) |
| *v1* | Pointer to primary color vector (input) |
| *p* | Interpolation value (input) |
| *v2* | Pointer to color vector (output) |

### Explanation

This function calculates the following:

LC = BK+LCM • v0

v2=p • v1 • LC + (1-p) • FC

v1 • LC is the product of multiplication.

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 19, 12) |
| *vl* -> r, g, b | : (0, 8, 0) |
| *p* | : (0, 20, 12) |
| *v2* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## ColorMatCol

Finds a color.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ColorMatCol** (**v0*, **v1*, **v2*, *matc*)
**SVECTOR** **v0*;
**CVECTOR** **v1*;
**CVECTOR** **v2*;
**long** *matc*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to normal vector (input) |
| *v1* | Pointer to primary color vector (input) |
| *v2* | Pointer to color vector (output) |
| *matc* | Material (input) |

### Explanation

This function performs the following calculations:

LLV = LLM • v0

LLV = LLV^ (2^matc)

LC = BK + LCM • LLV

v2 = v1 • LC (separate multiplications)

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 3, 12) |
| *v1* -> r, g, b | : (0, 8, 0) |
| *v2* -> r, g, b | : (0, 8, 0) |
| *matc* | : (0, 32, 0) |

### Return value

None.

### Remarks

**See also:**

## ColorMatDpq

Finds a color and performs depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ColorMatDpq** (**v0*, **v1*, *p*, **v2*, *matc*)
**SVECTOR** **v0*;
**CVECTOR** **v1*;
**long** *p*;
**CVECTOR** **v2*;
**long** *matc*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to normal vector (input) |
| *v1* | Pointer to primary color vector (input) |
| *p* | Interpolation value (output) |
| *v2* | Pointer to color vector (output) |
| *matc* | Material (output) |

### Explanation

This function performs the following calculations:

LLV = LLM • *v0*

LLV = LLV^ (2^*matc*)

LC = BK + LCM • LLV

*v2* = *p* • *v1* • LC + (1-p) • FC

v1*LC is the product of separate multiplications.

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 3, 12) |
| *v1* -> r, g, b | : (0, 8, 0) |
| *p* | : (0, 20, 12) |
| *v2* -> r, g, b | : (0, 8, 0) |
| *matc* | : (0, 32, 0) |

### Return value

None.

### Remarks

**See also:**

## CompMatrix

Make a composite coordinate transformation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX* CompMatrix** (**m0*, **m1*, **m2*)
**MATRIX** **m0*, **m1*, **m2*;

### Arguments

| | |
|---|---|
| *m0*, *m1* | Pointer to matrix (input) |
| *m2* | Pointer to matrix (output) |

### Explanation

This function makes a composite coordinate transformation matrix that includes parallel translation.

[m2 -> m] = [m0 -> m] • [m1 -> m]

(m2 -> t) = [m0 -> m] • (m1 -> t) + (m0 -> t)

However, the values of the elements of m1 -> t should be in the range (-2^15, 2^15).

Argument format

*m0* -> m[i][j] : (1, 3, 12)

*m0* -> t[i] : (1, 31, 0)

*m1* -> m[i][j] : (1, 3, 12)

*m1* -> t[i] : (1, 15, 0)

*m2* -> m[i][j] : (1, 3, 12)

*m2* -> t[i] : (1, 31, 0)

### Return value

*m2*

### Remarks

This function destroys a constant rotation matrix.

**See also:**

## csin

C sine function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long csin** (*a*)
**long** *a*;

### Arguments

| | |
|---|---|
| *a* | Angle (in Playstation format) |

### Explanation

Find the sine function of the angle (in Playstation format) (4096 = 360 degrees) using fixed point math (where 4096 = 1.0).

The argument format is as follows:

*a* : Playstation-format (4096 = 360 degrees)

Return value : (1, 19, 12)

### Return value

sin (a)

### Remarks

**See also:**

## csqrt

C square root function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long csqrt** (*a*)
**long** *a*;

### Arguments

| | |
|---|---|
| *a* | Value |

### Explanation

This function uses fixed point math (where 4096 = 1.0) to find the fixed point square root.

This function is the same as the SquareRoot12 function except that it requires a smaller table memory area.

The argument format is as follows:

*a*: (1, 19, 12)

Return value : (1, 19, 12)

### Return value

sqrt (*a*)

### Remarks

**See also:**

## DivideF3

Division of flat triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***DivideF3*** (**v0*, **v1*, **v2*, **rgbc*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*;
**CVECTOR** **rgbc*;
**POLY_F3** **s*;
**u_long** **ot*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vectors (input) |
| *rgbc* | Pointer to color vector + code (input) |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is a flat triangle division program. It divides a flat triangle (POLY_F3) indicated by the vertex coordinate vectors and color vector based on the *divp* -> ndiv value, and registers the result to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 7–3**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DivideF4

Division of flat quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***DivideF4** (**v0*, **v1*, **v2*, **v3*, **rgbc*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**CVECTOR** **rgbc*;
**POLY_F4** **s*;
**u_long** **ot*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vectors (input) |
| *rgbc* | Pointer to color vector + code (input) |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is a flat quadrilateral division program. It divides a flat quadrilateral (POLY_F4) indicated by the vertex coordinate vectors and color vector based on the divp -> ndiv value and registers the result to OT.

The divp -> ndiv values and division format are shown below:

**Table 7–4**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DivideFT3

Division of flat textured triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***DivideFT3** (**v0*, **v1*, **v2*, **uv0*, **uv1*, **uv2*, **rgbc*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*;
**u_long** **uv0*, **uv1*, **uv2*;
**CVECTOR** **rgbc*;
**POLY_FT3** **s*;
**u_long** **ot*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vectors (input) |
| *uv0*, *uv1*, *uv2* | Pointer to texture coordinate vector (input)<br>v0+clut, uv1:uv1+tpage (uv0) |
| *rgbc* | Pointer to color vector +code (input) |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is the flat textured triangle division program. It divides a flat textured triangle (POLY_FT3) indicated by the vertex coordinate vectors, texture coordinate vector, and color vector based on the *divp* -> ndiv value and registers the result to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 7–5**

| ndiv value | processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DivideFT4

Division of flat textured quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***DivideFT4** (**v0*, **v1*, **v2*, **v3*, **uv0*, **uv1*, **uv2*, **uv3*, **rgbc*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**u_long** **uv0*, **uv1*, **uv2*, **uv3*;
**CVECTOR** **rgbc*;
**POLY_FT4** **s*;
**u_long** **ot*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vectors (input) |
| *uv0*, *uv1*, *uv2*, *uv3* | Pointer to texture coordinate vector (input) uv0:uv0+clut, uv1:uv1+tpage |
| *rgbc* | Pointer to color vector + code (input) |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is the flat textured quadrilateral division program. It divides a flat textured quadrilateral (POLY_FT4) indicated by the vertex coordinate vectors, texture coordinate vector, and color vector based on the *divp* -> ndiv value and registers the result to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 7–6**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DivideG3

Division of Gouraud triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long *DivideG3** (**v0*, **v1*, **v2*, **rgb0*, **rgb1*, **rgb2*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*;
**POLY_G3** **s*;
**u_long** **ot*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vectors (input) |
| *rgb0*, *rgb1*, *rgb2* | Pointer to color vector (input)<br>rgb0:rgb0+code |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is a Gouraud-shaded triangle division program. It divides a Gouraud-shaded (POLY_G3) triangle indicated by the vertex coordinate vectors and color vector based on the *divp* -> ndiv value and registers the result to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 7–7**

| ndiv value | processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DivideG4

Division of Gouraud-shaded quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long *DivideG4** (**v0*, **v1*, **v2*, **v3*, **rgb0*, **rgb1*, **rgb2*, **rgb3*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*, **rgb3*;
**POLY_G4** **s*;
**u_long** **ot*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vectors (input) |
| rgb0, rgb1, rgb2, rgb3 | Pointer to color vector (input)<br>rgb0:rgb0+code |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is the Gouraud-shaded quadrilateral division program. It divides a Gouraud-shaded quadrilateral (POLY_G4) indicated by the vertex coordinate vectors and color vector based on the *divp* -> ndiv value and registers the result to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 7–8**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DivideGT3

Division of Gouraud-shaded, textured triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***DivideGT3** (**v0*, **v1*, **v2*, **uv0*, **uv1*, **uv2*, **rgb0*, **rgb1*, **rgb2*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*;
**u_long** **uv0*, **uv1*, **uv2*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*;
**POLY_GT3** **s*;
**u_long** **ot*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vectors (input) |
| *uv0*, *uv1*, *uv2* | Pointer to texture coordinate vector (input)<br>uv0:uv0+clut, uv1:uv1+tpage |
| *rgb0*, *rgb1*, *rgb2* | Pointer to color vector (input)<br>rgb0:rgb0+code |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is the Gouraud-shaded textured triangle division program. It divides a Gouraud-shaded textured triangle (POLY_GT3) indicated by the vertex coordinate vectors, texture coordinate vector, and color vector based on the *divp* -> ndiv value and registers the result to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 7–9**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DivideGT4

Division of Gouraud-shaded textured quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***DivideGT4** (**v0*, **v1*, **v2*, **v3*, **uv0*, **uv1*, **uv2*, **uv3*, **rgb0*, **rgb1*, **rgb2*, **rgb3*, **s*, **ot*, **divp*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**u_long** **uv0*, **uv1*, **uv2*, **uv3*;
**CVECTOR** **rgb0*, **rgb1*, **rgb2*, **rgb3*;
**POLY_GT4** **s*;
**u_long** **ot*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vertex coordinate vectors (input) |
| *uv0*, *uv1*, *uv2*, *uv3* | Pointer to texture coordinate vector (input)<br>uv0:uv0+clut, uv1:uv1+tpage |
| *rgb0*, *rgb1*, *rgb2*, *rgb3* | Pointer to color vector (input)<br>rgb0:rgb0+code |
| *s* | Pointer to GPU packet buffer address |
| *ot* | Pointer to OT entry |
| *divp* | Pointer to division work area (input) |

### Explanation

This is the Gouraud-shaded textured quadrilateral division program. It divides a Gouraud-shaded textured quadrilateral (POLY_GT4) indicated by the vertex coordinate vectors, texture coordinate vector, and color vector based on the *divp* -> ndiv value and registers the result to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 7–10**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

### Return value

Updated GPU packet buffer address.

### Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## DpqColor

Interpolation of a primary color vector and far color.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void DpqColor** (**v0*, *p*, **v1*)
**CVECTOR** **v0*;
**long** *p*;
**CVECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to primary color vector (input) |
| *p* | Interpolation value (input) |
| *v1* | Pointer to primary color vector (input) |

### Explanation

This function calculates *v1*=p • v0+ (1-p) • FC. The argument format is as follows:

| | |
|---|---|
| *v0* -> r, g, b | : (0, 8, 0) |
| *p* | : (0, 20, 12) |
| *v1* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## DpqColor3

Interpolation of three primary color vectors and far color.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void DpqColor3** (**v0*, **v1*, **v2*, *p*, **v3*, **v4*, **v5*)
**CVECTOR** **v0*, **v1*, **v2*;
**long** *p*;
**CVECTOR** **v3*, **v4*, **v5*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to primary color vectors (input) |
| *p* | Interpolation value (input) |
| *v3*, *v4*, *v5* | Pointer to color vectors (output) |

### Explanation

This function calculates:

*v3* = p • (v0)+ (1-p) • FC.

*v4* = p • (v1)+ (1-p) • FC.

*v5* = p • (v2)+ (1-p) • FC.

The argument format is follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> r, g, b | : (0, 8, 0) |
| *p* | : (0, 20, 12) |
| *v3*, *v4*, *v5* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## DpqColorLight

Interpolation of the product from the multiplication of a local color vector by primary color vector, and far color.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void DpqColorLight** (**v0*, **v1*, *p*, **v2*)
**SVECTOR** **v0*;
**CVECTOR** **v1*;
**long** *p*;
**CVECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to local color vector (input) |
| *v1* | Pointer to primary color vector (input) |
| *p* | Interpolation value (input) |
| *v2* | Pointer to color vector (output) |

### Explanation

This function calculates *v2* = p • (v1 • v0) + (1-p) • FC. *v1* • *v0* are separate multiplication products.

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 3, 12) |
| *v1* -> r, g, b | : (0, 8, 0) |
| *p* | : (0, 20, 12) |
| *v2* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## gteMIMefunc

Adding a vertex data array to a differential data array multiplied by a coefficient.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void gteMIMefunc** (**otp*, **dfp*, *n*, *p*)
**SVECTOR** **otp*;
**SVECTOR** **dfp*;
**long** *n*;
**long** *p*;

### Arguments

| | |
|---|---|
| *otp* | Pointer to a vertex array |
| *dfp* | Pointer to a differential array |
| *n* | Number of vertex (differential) data |
| *p* | Weight (control) coefficient: (1, 19, 12) |

### Explanation

Executes calculation of multiple interpolations using vertex data array and difference data array. The argument format is as follows.

*p*: (1, 19, 12)

*otp*, *dfp* optional

It operates at high speed in a similar way to the program given in the example below.

```
void gteMIMefunc (otp, dfp, n, p)
SVECTOR *otp, *dfp;
long n, p;
{
int i;
for (i = 0; i<n; i++) {
(otp+i)->x+=((int)((dfp+i)->x) • p)>>12;
(otp+i)->y+=((int)((dfp+i)->y) • p)>>12;
(otp+i)->z+=((int)((dfp+i)->z) • p)>>12;
    }
}
```

### Return value

None.

### Remarks

**See also:**

## InitClip

Initialize clipping parameter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void InitClip** (**evbfad*, *hw*, *vw*, *h*, *near*, *far*)
**EVECTOR** **evbfad*;
**long** *hw*, *vw*;
**long** *h*;
**long** *near*, *far*;

### Arguments

| | |
|---|---|
| *evbfad* | Pointer to addresses of (16) clip vector data arrays |
| *hw*, *vw* | *hw*: Window width, *vw*: Window height |
| *h* | Projection distance from view point to screen |
| *near*, *far* | *near*: NearClip position, *far*: FarClip position |

### Explanation

This function sets parameters used for clipping.

The clip vector data array *evbfad* reserves 16 data arrays (176 words or 704 bytes).

### Return value

None.

### Remarks

**See also:**

## InitGeom

Initialization of geometry transformation engine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void InitGeom** (*void*)

### Arguments

None.

### Explanation

This function initializes GTE. It is called when the basic geometry library is used.

### Return value

None.

### Remarks

**See also:**

## Intpl

Interpolation of a vector and far color.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void Intpl** (**v0*, *p*, **v1*)
**SVECTOR** **v0*;
**long** *p*;
**CVECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *p* | Interpolation value (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function calculates *v1* = p • v0 + (1-p) • FC.

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 3, 12) |
| *p* | : (0, 20, 12) |
| *v1* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## InvSquareRoot

1/square root.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void InvSquareRoot** (*a*, **b*, **c*)
**long** *a*;
**long** **b*;
**long** **c*;

### Arguments

| | |
|---|---|
| *a* | Value |
| *b* | Pointer to address where a mantissa will be stored |
| *c* | Pointer to address where an exponent will be stored |

### Explanation

The function returns 1/square root of a value *a*.

The argument format is as follows:

*a*: (0, 32, 0)

*b*: (0, 20, 12)

*c*: (0, 32, 0)

### Return value

None.

### Remarks

**See also:**

## LightColor

Coordinate transformation using the local color matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void LightColor** (**v0*, **v1*)
**SVECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function calculates v1=LCM • v0. A limiter works on negative components of *v1* when 0 is reached. The argument format is as follows:

*v0* -> vx, vy, vz : (1, 3, 12)

*v1* -> vx, vy, vz : (0, 20, 12)

### Return value

None.

### Remarks

**See also:**

## LoadAverage0

Weighted average of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void LoadAverage0** (**v0*, **v1*, *p0*, *p1*, **v2*)
**VECTOR** **v0*, **v1*;
**long** *p0*, *p1*;
**VECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0*, *v1* | Pointer to vectors (input) |
| *p0*, *p1* | Weights (input) |
| *v2* | Pointer to vector (output) |

### Explanation

This function returns the weighted average of two vectors *v0* and *v1* in *v2* using weights of *p0* and *p1*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1* -> vx, vy, vz | : (1, 31, 0) |
| *p0*, *p1* | : (1, 15, 0) |
| *v2* -> vx, vy, vz | : (1, 31, 0) |

### Return value

None.

### Remarks

**See also:**

## LoadAverage12

Weighted average of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void LoadAverage12** (**v0*, **v1*, *p0*, *p1*, **v2*)
**VECTOR** **v0*, **v1*;
**long** *p0*, *p1*;
**VECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0*, *v1* | Pointer to vectors (input) |
| *p0*, *p1* | Weights (input) |
| *v2* | Pointer to vector (output) |

### Explanation

This function finds the weighted average of two vectors *v0* and *v1* using weights of *p0* and *p1* after division by 4096 (1 in fixed point format) the results are returned in *v2*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1* -> vx, vy, vz | : (1, 31, 0) |
| *p0*, *p1* | : (1, 3, 12) |
| *v2* -> vx, vy, vz | : (1, 31, 0) |

### Return value

None.

### Remarks

**See also:**

## LoadAverageByte

Find weighted average of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void LoadAverageByte** (*v0*, *v1*, *p0*, *p1*, *v2*)
**unsigned char** *v0*[2], *v1*[2];
**long** *p0*, *p1*;
**unsigned char** *v2*[2];

### Arguments

| | |
|---|---|
| *v0*, *v1* | Vector (input) |
| *p0*, *p1* | Weights (input) |
| *v2* | Vector (output) |

### Explanation

This function finds the weighted average of two vectors *v0* and *v1* using weights *p0* and *p1*. The result is returned in *v2* after division by 4096.

The argument format is as follows:

| | |
|---|---|
| *v0*[i], *v1*[i] | : (0, 8, 0) |
| *p0*, *p1* | : (1, 3, 12) |
| *v2*[i] | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## LoadAverageCol

Find weighted average of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void LoadAverageCol** (*v0*, *v1*, *p0*, *p1*, *v2*)
**unsigned char** *v0*[3], *v1*[3];
**long** *p0*, *p1*;
**unsigned char** *v2*[3];

### Arguments

| | |
|---|---|
| *v0*, *v1* | Vectors (input) |
| *p0*, *p1* | Weights (input) |
| *v2* | Vector (output) |

### Explanation

This function finds the weighted average of two vectors *v0* and *v1* using weights *p0* and *p1*. The result is returned in *v2* after division by 4096.

The argument format is as follows:

| | |
|---|---|
| *v0*[i], *v1*[i] | : (0, 8, 0) |
| *p0*, *p1* | : (1, 3, 12) |
| *v2*[i] | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## LoadAverageShort0

Weighted average of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void LoadAverageShort0** (**v0*, **v1*, *p0*, *p1*, **v2*)
**SVECTOR** **v0*, **v1*;
**long** *p0*, *p1*;
**SVECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0*, *v1* | Pointer to vectors (input) |
| *p0*, *p1* | Weights (input) |
| *v2* | Pointer to vector (output) |

### Explanation

This function returns the weighted average of two vectors *v0* and *v1* in *v2* using weights of *p0* and *p1*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1* -> vx, vy, vz | : (1, 15, 0) |
| *p0*, *p1* | : (1, 15, 0) |
| *v2* -> vx, vy, vz | : (1, 30, 0) |

### Return value

None.

### Remarks

**See also:**

## LoadAverageShort12

Weighted average of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void LoadAverageShort12** (**v0*, **v1*, *p0*, *p1*, **v2*)
**SVECTOR** **v0*, **v1*;
**long** *p0*, *p1*;
**SVECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0*, *v1* | Pointer to vectors (input) |
| *p0*, *p1* | Weights (input) |
| *v2* | Pointer to vector (output) |

### Explanation

This function finds the weighted average of two vectors *v0* and *v1* using weights of *p0* and *p1* after division by 4096 (1 in fixed point format) the results are returned to *v2*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1* -> vx, vy, vz | : (1, 15, 0) |
| *p0*, *p1* | : (1, 3, 12) |
| *v2* -> vx, vy, vz | : (1, 15, 0) |

### Return value

None.

### Remarks

**See also:**

## LocalLight

Coordinate transformation using the local light matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void LocalLight** (**v0*, **v1*)
**SVECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function calculates *v1*=LLM*v0. A limiter works on negative components of *v1* when 0 is reached. The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz: | :(1, 3, 12) |
| *v1* -> vx, vy, vz: | :(0, 20, 12) |

### Return value

None.

### Remarks

**See also:**

## Lzc

Returning a leading zero count (LZC).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long Lzc** (*data*)
**long data**;

### Arguments

| | |
|---|---|
| *data* | Value |

### Explanation

This function calculates the leading zero count given by *data*. In short, when the data is displayed as binary, a value identical to MSB returns a number of bits from MSB.

The argument format is as follows:

| | |
|---|---|
| *data* | : (1, 31, 0) |
| Return value | : (1, 31, 0) |

### Return value

Returns the value of LZC.

### Remarks

**See also:**

## MatrixNormal

Normalize a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void MatrixNormal** (**m*, **n*)
**MATRIX** **m*;
**MATRIX** **n*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |
| *n* | Pointer to matrix (output) |

### Explanation

This function orthogonalizes and normalizes a rotation-matrix m and returns the result in n.

**Note:**

This function doesn't use m->m[2][0],m->m[2][1],m->m[2][2]. The argument format is as follows:

m->m[i][j]:(1.3.12)
n->m[i][j]:(1.3.12)

### Return value

### Remarks

**See also:**

## MatrixNormal_1

Normalize a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void MatrixNormal_1** (**m*, **n*)
**MATRIX** **m*;
**MATRIX** **n*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |
| *n* | Pointer to matrix (output) |

### Explanation

This function orthogonalizes and normalizes a rotation-matrix m and returns the result in n.

**Note:**

This function doesn't use m->m[0][0],m->m[0][1],m->m[0][2]. The argument format is as follows:

m->m[i][j]:(1.3.12)
n->m[i][j]:(1.3.12)

### Return value

### Remarks

**See also:**

## MatrixNormal_2

Normalize a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void MatrixNormal_2** (**m*, **n*)
**MATRIX** **m*;
**MATRIX** **n*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |
| *n* | Pointer to matrix (output) |

### Explanation

This function orthogonalizes and normalizes a rotation-matrix m and returns the result in n.

**Note:**

This function doesn't use m->m[1][0],m->m[1][1],m->m[1][2]. The argument format is as follows:

m->m[i][j]:(1.3.12)
n->m[i][j]:(1.3.12)

### Return value

### Remarks

**See also:**

## MulMatrix

Multiplication of two matrices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX** ***MulMatrix*** (**m0*, **m1*)
**MATRIX** **m0*, **m1*;

### Arguments

| | |
|---|---|
| *m0*, *m1* | Pointer to input/output matrices |

### Explanation

This function multiplies two matrices. The result is saved in *m0*. The argument format is as follows:

*m0*, *m1* -> m[i][j]: (1, 3, 12)

### Return value

This function returns *m0*.

### Remarks

The function destroys the constant rotation matrix.

**See also:**

## MulMatrix0

Multiplication of two matrices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX** ***MulMatrix0** (**m0*, **m1*, **m2*)
**MATRIX** **m0*, **m1*;
**MATRIX** **m2*;

### Arguments

| | |
|---|---|
| *m0*, *m1* | Pointer to input matrices |
| *m2* | Pointer to output matrix |

### Explanation

This function multiplies two matrices *m0* and *m1*.

The argument format is as follows:

*m0*, *m1*, *m2* -> m[i][j] : (1, 3, 12)

### Return value

This function returns *m2*.

### Remarks

The function destroys the constant rotation matrix.

**See also:**

## MulMatrix2

Multiplication of two matrices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX** ***MulMatrix2** (**m0*, **m1*)
**MATRIX** **m0*, **m1*;

### Arguments

| | |
|---|---|
| *m0*, *m1* | Pointer to input/output matrices |

### Explanation

This function multiplies two matrices. The result is saved in *m1*. The argument format is as follows:

*m0*, *m1* -> m[i][j]: (1, 3, 12)

### Return value

This function returns *m1*.

### Remarks

The function destroys the constant rotation matrix.

**See also:**

## MulRotMatrix

Multiply a constant rotation matrix by a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX*** **MulRotMatrix** (**m0*)
**MATRIX** *m0;

### Arguments

*m0* Pointer to input/output matrix

### Explanation

This function multiplies a constant rotation matrix by a matrix. It stores the value in *m0*.

The argument format is as follows:

*m0*, *m1* -> m[i][j] : (1, 3, 12)

### Return value

Returns *m0*.

### Remarks

**See also:**

## MulRotMatrix0

Multiply a constant rotation matrix by a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX*** **MulRotMatrix0** (**m0*, **m1*)
**MATRIX** *m0;
**MATRIX** *m1;

### Arguments

| | |
|---|---|
| *m0* | Pointer to input matrix |
| *m1* | Pointer to output matrix |

### Explanation

This function multiplies a constant rotation matrix by matrix *m0*. The result is saved in *m1*.

The argument format is as follows:

*m0*, *m1* -> m[i][j] : (1, 3, 12)

### Return value

Returns *m1.*

### Remarks

**See also:**

## NormalClip

Outer product of three points.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long NormalClip** (*sxy0*, *sxy1*, *sxy2*)
**long** *sxy0*, *sxy1*, *sxy2*;

### Arguments

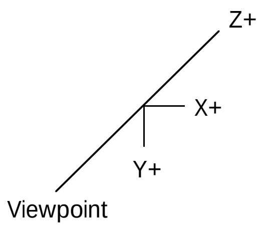
| | |
|---|---|
| *sxy0*, *sxy1*, *sxy2* | Vertex coordinates (upper position 16-bit is y coordinate and lower position 16-bit is x coordinate) |

### Explanation

This function returns the outer product for a triangle formed by three points (sx0, sy0), (sx1, sy1), and (sx2, sy2). Example:

If the triangle is defined clockwise as seen from the visual point:

Figure 7–1



The argument format is as follows:

*sxy0*, *sxy1*, *sxy2*: y (1, 15, 0), x (1, 15, 0)

### Return value

The function returns the outer product for the triangle formed by three points (sx0, sy0), (sx1, sy1), and (sx2, sy2).

$$\begin{vmatrix} sx1-sx0, & sy1-sy0 \\ sx2-sx0, & sy2-sy0 \end{vmatrix}$$

### Remarks

**See also:**

## NormalColor

Finds a local color from a normal vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void NormalColor** (**v0*, **v1*)
**SVECTOR** **v0*;
**CVECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to normal vector (input) |
| *v1* | Pointer to color vector (output) |

### Explanation

This function calculates LLV=LLM • v0, v1=BK+LCM • LLV. The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 3, 12) |
| *vl* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## NormalColor3

Finds three local colors from three normal vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void NormalColor3** (**v0*, **v1*, **v2*, **v3*, **v4*, **v5*)
**SVECTOR** **v0*, **v1*, **v2*;
**CVECTOR** **v3*, **v4*, **v5*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to normal vectors (input) |
| *v3*, *v4*, *v5* | Pointer to color vectors (output) |

### Explanation

This function calculates

(LLV0, LLV1, LLV2) = LLM • (v0, v1, v2)

(v3, v4, v5) = BK +LCM • (LLV0, LLV1, LLV2)

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 3, 12) |
| *v3*, *v4*, *v5* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## NormalColorCol

Finds a local color from a normal vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void NormalColorCol** (**v0*, **v1*, **v2*)
**SVECTOR** **v0*;
**CVECTOR** **v1*;
**CVECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to normal vector (input) |
| *v1* | Pointer to primary color vector (input) |
| *v2* | Pointer to color vector (output) |

### Explanation

This function calculates the following:

LLV=LLM • v0

LC = BK + LCM • LLV

v2 = v1 • LC (separate multiplication)

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 3, 12) |
| *v1* -> r, g, b | : (0, 8, 0) |
| *v2* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## NormalColorCol3

Finds a local color from three normal vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void NormalColorCol3** (**v0*, **v1*, **v2*, **v3*, **v4*, **v5*, **v6*)
**SVECTOR** **v0*, **v1*, **v2*;
**CVECTOR** **v3*;
**CVECTOR** **v4*, **v5*, **v6*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to normal vectors |
| *v3* | Pointer to primary color vector (input) |
| *v4*, *v5*, *v6* | Pointer to color vectors (output) |

### Explanation

This function calculates the following:

(LLV0, LLV1, VVL2)=LLM • (v0, v1, v2)

(LC0, LC1, LC2) = BK + LCM • (LLV0, LLV1, LLV2)

(v4, v5, v6) = v3 • (LC0, LC1, LC2) (separate multiplication)

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 3, 12) |
| *v3* -> r, g, b | : (0, 8, 0) |
| *v4*, *v5*, *v6* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## NormalColorDpq

Finds a local color from a normal vector and performs depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void NormalColorDpq** (**v0*, **v1*, *p*, **v2*)
**SVECTOR** **v0*;
**CVECTOR** **v1*;
**long** *p*;
**CVECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to normal vector (input) |
| *v1* | Pointer to primary color vector (input) |
| *p* | Interpolation value (input) |
| *v2* | Pointer to color vector (output) |

### Explanation

This function calculates the following:

LLV=LLM • *v0*

LC = BK +LCM • LLV

v2 = (1-p) • (v1 • LC)+p • FC

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 3, 12) |
| *vl* -> r, g, b | : (0, 8, 0) |
| *p* | : (0, 20, 12) |
| *v2* -> r, g, b | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## NormalColorDpq3

Finds local color from three normal vectors, and performs depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void NormalColorDpq3** (**v0*, **v1*, **v2*, **v3*, *p*, **v4*, **v5*, **v6*)
**SVECTOR** **v0*, **v1*, **v2*;
**CVECTOR** **v3*;
**long** *p*;
**CVECTOR** **v4*, **v5*, **v6*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to normal vectors (input) |
| *v3* | Pointer to primary color vector (input) |
| *p* | Interpolation value (input) |
| *v4*, *v5*, *v6* | Pointer to color vectors (output) |

### Explanation

This function calculates the following:

(LLV0, LLV1, LLV2) = LLM • (v0, v1, v2)

(LC0, LC1, LC2) = BK + LCM • (LLV0, LLV1, LLV2)

(v4, v5, v6) = p • (v3 • (LC0, LC1, LC2)) + (1 -p) • FC

v3 • (LC0, LC1, LC2) is the product of separate multiplications.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 3, 12) |
| *v3* -> r, g, b | : (0, 8, 0) |
| *p* | : (0, 20, 12) |
| *v4*, *v5*, *v6* -> r, g, b) | : (0, 8, 0) |

### Return value

None.

### Remarks

**See also:**

## otz2p

Get depth cueing interpolation value p from OTZ value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long otz2p (**
**long** *otz*,
**long** *projection*
**)**

### Arguments

| | |
|---|---|
| *otz* | OTZ |
| *projection* | Distance between visual point and screen |

### Explanation

Get the approximate depth cueing interpolation value p from sz, the z element of the screen coordinates. sz is sz/4 *otz* value.

### Return value

Depth cueing interpolation value p (0: 0%, 4096 : 100%).

### Remarks

Depending on the fog setting, errors can increase and the results are not necessarily the same as the RotTransPers system function.

**See also:**

## OuterProduct0

Outer product of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void OuterProduct0** (**v0*, **v1*, **v2*)
**VECTOR** **v0*, **v1*;
**VECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0*, *v1* | Pointer to vectors (input) |
| *v2* | Pointer to vector (output) |

### Explanation

This function returns the outer product vector of two vectors *v0* and *v1* to *v2*. The argument format is as follows:

| | |
|---|---|
| *v0*, *v1* -> vx, vy, vz | : (1, 31, 0) |
| *v2* -> vx, vy, vz | : (1, 31, 0) |

### Return value

None.

### Remarks

**See also:**

## OuterProduct12

Outer product of two vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void OuterProduct12** (**v0*, **v1*, **v2*)
**VECTOR** **v0*, **v1*;
**VECTOR** **v2*;

### Arguments

| | |
|---|---|
| *v0*, *v1* | Pointer to vectors (input) |
| *v2* | Pointer to vector (output) |

### Explanation

This function returns the outer product vector of two vectors, *v0* and *v1,* to *v2*. The argument format is as follows:

*v0*, *v1*, *v2* -> vx, vy, vz: :(1, 19, 12)

### Return value

None.

### Remarks

**See also:**

## p2otz

Get otz from depth cueing interpolation value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long p2otz (**
**long** *p*,
**long** *projection*
**)**

### Arguments

| | |
|---|---|
| *p* | Can be 0 to 4096 |
| *projection* | Distance between visual point and screen |

### Explanation

Gets the z element of the screen coordinates or sz/4 *otz* value from the depth cueing interpolation value *p*.

### Return value

OTZ value.

### Remarks

Depending on the fog setting, errors can increase and the results are not necessarily the same as the RotTransPers system functions.

*otz* when P=0 or p=4096 is not theoretically decided as identification, but with this function a convenient value is returned.

**See also:**

## pers_map

Perspective conversion texture mapping.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void pers_map (**
**int** *abuf*,
**SVECTOR** ***vertex*,
**int** *tex*[4][2],
**u_short** **dtext*

### Arguments

| | |
|---|---|
| *abuf* | ID of displayed buffer |
| *vertex* | 3 dimensional coordinates of 4 vertices |
| *tex* | Texture address of 4 vertices |
| *dtext* | Pointer to texture storage location converted to direct color |

### Explanation

Performs texture mapping with no distortion.

### Return value

None

### Remarks

Flat texture, with no light source calculations only.

The 4 vertices are only square, rectangle and parallelogram locations.

Z sort by OT is not possible.

**See also:**

## PhongLine

Phong shading.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void PhongLine(**
**int** *istart_x*,
**int** *iend_x*,
**int** *p*,
**int** *q*,
**u_short** **pixx*,
**int** *fs*,
**int** *ft*,
**int** *i4*,
**int** *det*
**)**

### Arguments

| | |
|---|---|
| *isstat_x* | X coordinate of starting point |
| *iend_x* | X coordinate of finishing point |
| *p* | Differential X coordinate of fs value |
| *q* | Differential caused by X coordinate of ft value |
| *pixx* | Pixel pointer |
| *fs* | Interpolation coefficient at start point |
| *ft* | Interpolation coefficient at start point |
| *i4* | (Line number) %4 due to 'Dithering' |
| *det* | Queue method of edge queue |

### Explanation

Performs one line Phong shading.

### Return value

None

### Remarks

Refer to sample program (sample/graphics/phong)

**See also:**

## PopMatrix

Resets a constant rotation matrix from a stack.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void PopMatrix** (*void*)

### Arguments

None.

### Explanation

This function resets a constant rotation matrix from a stack.

### Return value

None.

### Remarks

**See also:**

## PushMatrix

Saving a constant rotation matrix in a stack.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void PushMatrix** (*void*)

### Arguments

None.

### Explanation

This function saves a constant rotation matrix on a stack. The stack has 20 slots.

### Return value

None.

### Remarks

**See also:**

## ratan2

Arctan.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long ratan2** (*y*, *x*)
**long** *y*, *x*;

### Arguments

| | |
|---|---|
| *y*, *x* | Value |

### Explanation

This function uses Playstation format (4096 = 360 degrees) to finish the y/x arctan function (-180 degrees and +180 degrees).

The argument format is as follows:

(x, y) : (1, 31, 0)

Return value : Playstation format (4096 = 360 degrees)

### Return value

This function returns the *y*/*x* arctan function (atan2 (y,x)).

### Remarks

The return value is incorrect if either x or y is –2147483648 (0x80000000 = long negative's maximum value).

**See also:**

## rcos

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long rcos** (*a*)
**long int** *a*;

### Arguments

| | |
|---|---|
| *a* | Angle (in Playstation format) |

### Explanation

Finds the cosine function of the angle (in Playstation format) (4096=360 degrees) using fixed-point math (where 4096=1.0).

The argument format is as follows:

*a* : Playstation format (4096=360 degrees)

Return value : (1, 19, 12)

### Return value

cos (a)

### Remarks

**See also:**

## RCpolyF3

Division of flat triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***RCpolyF3** (**s*, **divp*)
**POLY_F3** **s*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of flat triangles (POLY_F3). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2* | Division vertex vector data |
| CRVECTOR3 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR3 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*, and *r2*.

Note: See DIVPOLYGON3 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON3 (p. 7-10).

## RCpolyF4

Division of flat quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***RCpolyF4** (**s*, **divp*)
**POLY_F4** **s*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of flat quadrilaterals (POLY_F4). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2*, *r3* | Division vertex vector data |
| CRVECTOR4 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR4 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*, *r2* and *r3*.

Note: See DIVPOLYGON4 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON4 (p. 7-11).

## RCpolyFT3

Division of flat, textured triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** *****RCpolyFT3** (**s*, **divp*)
**POLY_FT3** **s*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of flat , textured triangles (POLY_FT3). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2* | Division vertex vector data |
| CRVECTOR3 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR3 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*,and *r2*.

Note: See DIVPOLYGON3 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON3 (p. 7-10).

## RCpolyFT4

Division of flat, textured quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***RCpolyFT4** (**s*, **divp*)
**POLY_FT4** **s*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of flat, textured quadrilaterals (POLY_FT4). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2*, *r3* | Division vertex vector data |
| CRVECTOR4 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR4 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*, *r2* and *r3*.

Note: See DIVPOLYGON4 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON4 (p. 7-11).

## RCpolyG3

Division of Gouraud-shaded triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***RCpolyG3*** (**s*, **divp*)
**POLY_G3** **s*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of Gourard-shaded triangles (POLY_G3). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2* | Division vertex vector data |
| CRVECTOR3 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR3 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*,and *r2*.

Note: See DIVPOLYGON3 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON3 (p. 7-10).

## RCpolyG4

Division of Gouraud-shaded quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***RCpolyG4** (**s*, **divp*)
**POLY_G4** **s*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of Gouraud-shaded quadrilaterals (POLY_G4). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2*, *r3* | Division vertex vector data |
| CRVECTOR4 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR4 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*, *r2* and *r3*.

Note: See DIVPOLYGON4 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON4 (p. 7-11).

## RCpolyGT3

Division of Gouraud-shaeded, textured triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***RCpolyGT3** (**s*, **divp*)
**POLY_GT3** **s*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of Gourard-shaded, textured triangles (POLY_GT3). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2* | Division vertex vector data |
| CRVECTOR3 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR3 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*,and *r2*.

Note: See DIVPOLYGON3 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON3 (p. 7-10).

## RCpolyGT4

Division of Gouraud-shaded, textured quadrilateral.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***RCpolyGT4** (**s*, **divp*)
**POLY_GT4** **s*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *s* | Pointer to GPU packet buffer address |
| *divp* | Pointer to division work area |

### Explanation

This is a recursive function for division of Gouraud-shaded, textured quadrilaterals (POLY_GT4). In order to use it, you must set the data below in the *divp* work area:

| | |
|---|---|
| u_long *ndiv* | Number of divisions |
| u_long *pih*, *piv* | Display screen resolution (for clipping) |
| u_short *clut*, *tpage* | CBA & TSB |
| CVECTOR *rgbc* | Color vector (+code) |
| u_long **ot* | OT entry |
| RVECTOR *r0*, *r1*, *r2*, *r3* | Division vertex vector data |
| CRVECTOR4 *cr*[5]; | 2D and 3D texture coordinates and color for each vertex |

Assign the vertex vector data of CRVECTOR4 (*cr*[5]) to the value of the vertex vector data of RVECTOR's *r0*, *r1*, *r2* and *r3*.

Note: See DIVPOLYGON4 for a full description of *divp*.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:** DIVPOLYGON4 (p. 7-11).

## ReadColorMatrix

Reading a local color matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ReadColorMatrix** (**m*)
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |

### Explanation

This function reads the current local color matrix, and saves it in *m*.

The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

### Return value

None.

### Remarks

**See also:**

## ReadGeomOffset

Read GTE offset value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void ReadGeomOffset** (**ofx*, **ofy*)
**long** **ofx*, **ofy*;

### Arguments

| | |
|---|---|
| *ofx* | Pointer to offset X coordinate |
| *ofy* | Pointer to offset Y coordinate |

### Explanation

This function reads the GTE offset value.

The argument format is as follows:

ofx, ofy : (0, 32, 0)

### Return value

None.

### Remarks

**See also:**

## ReadGeomScreen

Read distance from view point to screen.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long ReadGeomScreen** (*void*)

### Arguments

None.

### Explanation

This function reads the distance h from the view point (eye) to the screen.

The argument format is as follows:

Return value : (0, 32, 0)

### Return value

h value

### Remarks

**See also:**

## ReadLightMatrix

Reading a local light matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ReadLightMatrix** (**m*)
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |

### Explanation

This function reads the current local light matrix, and saves it in *m*.

The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

### Return value

None.

### Remarks

**See also:**

## ReadRGBfifo

Reading RGBcd values.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ReadRGBfifo** (**v0*, **vl*, **v2*)
    **CVECTOR** **v0*, **v1*, **v2*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vectors (output) |

### Explanation

This function stores the RGBcd0, RGBcd1, and RGBcd2 values in *v0*, *v1*, and *v2*. The argument format is as follows:

*v0*, *v1*, *v2* -> r, g, b, cd: (0, 8, 0)

### Return value

None.

### Remarks

**See also:**

## ReadRotMatrix

Reads a constant rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ReadRotMatrix** (**m*)
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (output) |

### Explanation

This function reads the current rotation matrix, and saves it in *m*. The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

*m* -> t[i]: (1, 31, 0)

### Return value

None.

### Remarks

**See also:**

## ReadSXSYfifo

Reads SXSY values.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ReadSXSYfifo** (**sxy0*, **sxy1*, **sxy2*)
**long** **sxy0*, **sxy1*, **sxy2*;

### Arguments

| | |
|---|---|
| *sxy0*, *sxy1*, *sxy2* | Pointer to addresses where SZ values are stored |

### Explanation

This function stores the sx0, sy0, sx1, sy1, sx2, and sy2 values in *sxy0*, *sxy1*, and *sxy2*. The argument format is as follows:

(*sxy0*, *sxy1*, *sxy2*): (1, 15, 0)

### Return value

None.

### Remarks

**See also:**

## ReadSZfifo3

Reads SZ values.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ReadSZfifo3** (**sz0*, **sz1*, **sz2*)
**long** **sz0*, **sz1*, **sz2*;

### Arguments

| | |
|---|---|
| *sz0*, *sz1*, *sz2* | Pointer to addresses where SZ values are stored |

### Explanation

This function stores the SZ0, SZ1, and SZ2 values in *sz0*, *sz1*, and *sz2*. The argument format is as follows:

(sz0, sz1, sz2): (0, 16, 0)

### Return value

None.

### Remarks

**See also:**

## ReadSZfifo4

Reads SZ values.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void ReadSZfifo4** (**szx*, **sz0*, **sz1*, **sz2*)
**long** **szx*, **sz0*, **sz1*, **sz2*;

### Arguments

| | |
|---|---|
| *szx*, *sz0*, *sz1*, *sz2* | Pointer to addresses where SZ values are stored |

### Explanation

This function stores the SZX, SZ0, SZ1, and SZ2 values in *szx*, *sz0*, *sz1*, and *sz2*. The argument format is as follows:

(*szx*, *sz0*, *sz1*, *sz2*): (0, 16, 0)

### Return value

None.

### Remarks

**See also:**

## RotAverage3

Perform coordinate transformation for 3 points and perspective transformation, and find an interpolation value and an average of Z values for depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotAverage3** (**v0*, **v1*, **v2*, **sxy0*, **sxy1*, **sxy2*, **p*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**long** **sxy0*, **sxy1*, **sxy2*;
**long** **p*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vectors (input) |
| *sxy0*, *sxy1*, *sxy2* | Pointer to address where the coordinates will be stored |
| *p* | Pointer to address where the interpolation value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of three points *v0*, *v1*, *v2* is performed using a rotation matrix. Next a perspective transformation is performed and three screen coordinates *sxy0*, *sxy1*, and *sxy2* are returned. An interpolation value for depth cueing on *v2* to *p* is also returned.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *flag* | : (0, 32, 0) |
| Return value | : (0, 32, 0) |

### Return value

1/4 (OTZ value) average of three screen coordinate Z values.

### Remarks

**See also:**

## RotAverage4

Perform coordinate transformation for 3 points and perspective transformation, and find an interpolation value and an average of Z values for depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotAverage4** (**v0*, **v1*, **v2*, **v3*, **sxy0*, **sxy1*, **sxy2*, **sxy3*, **p*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**long** **sxy0*, **sxy1*, **sxy2*, **sxy3*;
**long** **p*;
**long** **flag*;

### Arguments

| | |
|---|---|
| v0, v1, v2, v3 | Pointer to vectors (input) |
| sxy0, sxy1, sxy2, sxy3 | Pointer to address where the coordinates will be stored |
| p | Pointer to address where the interpolation value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of four points *v0*, *v1*, *v2* and *v3* is performed using a rotation matrix. Next a perspective transformation is performed and four screen coordinates *sxy0*, *sxy1*, sxy2, and *sxy3* are returned. An interpolation value for depth cueing on *v2* to *p* is also returned.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2*, *sxy3* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *flag* | : (0, 32, 0) |

### Return value

1/4 (OTZ value) average of four screen coordinate Z values.

### Remarks

**See also:**

## RotAverageNclip3

Perform coordinate transformation and perspective transformation for three points, and find an interpolation value, average of Z values, and outer product.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotAverageNclip3** (**v0*, **v1*, **v2*, **sxy0*, **sxy1*, **sxy2*, **p*, **otz*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**long** **sxy0*, **sxy1*, **sxy2*;
**long** **p*;
**long** **otz*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vectors (input) |
| *sxy0*, *sxy1*, *sxy2* | Pointer to address where coordinates will be stored |
| *p* | Pointer to address where an interpolation value will be stored |
| *otz* | Pointer to address where an OTZ value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of three points *v0*, *v1*, *v2* is performed using a rotation matrix. Next a perspective transformation is performed and three screen coordinates *sxy0*, *sxy1*, and *sxy2* are returned. An interpolation value for depth cueing on *v2* to *p* is also returned. Finally, we also receive 1/4 of the Z value of the screen coordinates for *v2* to *otz*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *otz* | : (0, 32, 0) |
| *flag* | : (0, 32, 0) |

### Return value

Outer product of (sx0, sy0), (sx1, sy1), (sx2, sy2).

### Remarks

When the return value is negative, SX, SY, etc. are incorrect. When SX and SY are required, use RotAverage3().

**See also:**

## RotAverageNclip4

Performs a coordinate transformation and perspective transformation for four points; finds an interpolation value, average of Z values, and outer product.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotAverageNclip4** (**v0*, **v1*, **v2*, **v3*, **sxy0*, **sxy1*, **sxy2*, **sxy3*, **p*, **otz*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**long** **sxy0*, **sxy1*, **sxy2*, **sxy3*;
**long** **p*;
**long** **otz*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vectors (input) |
| *sxy0*, *sxy1*, *sxy2*, *sxy3* | Pointer to address where coordinates will be stored |
| *p* | Pointer to address where an interpolation value will be stored |
| *otz* | Pointer to address where an OTZ value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of four points *v0*, *v1*, *v2*, and *v3* is performed using a rotation matrix. Next a perspective transformation is performed and four screen coordinates *sxy0*, *sxy1*, *sxy2* and *sxy3* are returned. An interpolation value for depth cueing on *v2* to *p* is also returned. Finally, we also receive 1/4 of the Z value of the screen coordinates for *v2* to *otz*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2*, *sxy3* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *otz* | : (0, 32, 0) |
| *flag* | : (0, 32, 0) |

### Return value

Outer product of (sx0, sy0), (sx1, sy1), (sx2, sy2).

### Remarks

When the return value is negative, SX, SY, etc., are incorrect. When SX and SY are required, use RotAverage4().

**See also:**

## RotAverageNclipColorCol3

Performs a coordinate transformation for three points, perspective transformation, and finds a color.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotAverageNclipColorCol3** (**v0*, **v1*, **v2*, **v3*, **v4*, **v5*, **v6*, **sxy0*, **sxy1*, **sxy2*, **v7*, **v8*, **v9*, **otz*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**SVECTOR** **v3*, **v4*, **v5*;
**CVECTOR** **v6*;
**long** **sxy0*, **sxy1*, **sxy2*;
**CVECTOR** **v7*, **v8*, **v9*;
**long** **otz*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vectors (input) |
| *v3*, *v4*, *v5* | Pointer to normal vectors (input) |
| *v6* | Pointer to primary color vector (input) |
| *sxy0*, s*xy1*, *sxy2* | Pointer to address where coordinate values will be stored |
| *v7*, *v8*, *v9* | Pointer to color vectors (output) |
| *otz* | Pointer to address where an OTZ value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of three points *v0*, *v1*, *v2* is performed using a rotation matrix. Next a perspective transformation is performed and four screen coordinates *sxy0*, *sxy1*, *sxy2* are returned. The remaining values are calculated as follows:

(LLV0, LLV1, LLV2) = LLM • (v3, v4, v5)

(LC0, LC1, LC2) = BK + LCM • (LLV0, LLV1, LLV2)

(v7, v8, v9) = v6 • (LC0, LC1, LC2)
(separate multiplications)

The function also returns an average of Z values of three screen coordinates to *otz*. The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 15, 0) |
| *v3*, *v4*, *v5* -> vx, vy, vz | : (1, 3, 12) |
| *v6* -> r, g, b | : (0, 8, 0) |
| *sxy0*, *sxy1*, *sxy2* | : (1, 15, 0), (1, 15, 0) |
| *v7*, *v8*, *v9* -> r, g, b | : (0, 8, 0) |
| *otz* | : (0, 32, 0) |
| *flag* | : (0, 32, 0) |

### Return value

Outer product of (sx0, sy0), (sx1, sy1), (sx2, sy2)

**Remarks**

When the return value is negative, SX, SY, etc., are incorrect.

**See also:**

## RotAverageNclipColorDpq3

Coordinate transformation for three points, perspective transformation, and depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotAverageNclipColorDpq3** (**v0*, **v1*, **v2*, **v3*, **v4*, **v5*, **v6*, **sxy0*, **sxy1*, **sxy2*, **v7*, **v8*, **v9*, **otz*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**SVECTOR** **v3*, **v4*, **v5*;
**CVECTOR** **v6*;
**long** **sxy0*, **sxy1*, **sxy2*;
**CVECTOR** **v7*, **v8*, **v9*;
**long** **otz*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vectors (input) |
| *v3*, *v4*, *v5* | Pointer to normal vectors (input) |
| *v6* | Pointer to primary color vector (input) |
| *sxy0*, *sxy1*, *sxy2* | Pointer to address where coordinate values will be stored |
| *v7*, *v8*, *v9* | Pointer to color vectors (output) |
| *otz* | Pointer to address where an OTZ value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of three points *v0*, *v1*, *v2* is performed using a rotation matrix. Next a perspective transformation is performed and three screen coordinates *sxy0*, *sxy1*, and *sxy2* are returned. The function uses the interpolation value *p* for depth cueing; *p* is found by the following calculations:

(LLV0, LLV1, LLV2) = LLM • (v3, v4, v5)

(LC0, LC1, LC2) = BK + LCM • (LLV0, LLV1, LLV2)

(v7, v8, v9) = p • (v6 • (LC0, LC1, LC2) + (1-p)) • FC

VC • ( LC0, LC1, LC2) indicates the products of separate multiplications.

The function also returns an average of the Z values of the three screen coordinates to otz. The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 15, 0) |
| *v3*, *v4*, *v5* -> vx, vy, vz | : (1, 3, 12) |
| *v6* -> r, g, b | : (0, 8, 0) |
| *sxy0*, *sxy1*, *sxy2* | : (1, 15, 0), (1, 15, 0) |
| *v7*, *v8*, *v9* -> r, g, b | : (0, 8, 0) |
| *otz* | : (0, 32, 0) |
| *flag* | : (0, 32, 0) |

### Return value

Outer product of (sx0, sy0), (sx1, sy1), (sx2, sy2)

**Remarks**

When the return value is negative, SX, SY, etc. are incorrect.

**See also:**

## RotColorDpq

Coordinate transformation for one point, perspective transformation, and depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotColorDpq** (**v0*, **v1*, **v2*, **sxy*, **v3*, **flag*)
**SVECTOR** **v0*;
**SVECTOR** **v1*;
**CVECTOR** **v2*;
**long** **sxy*;
**CVECTOR** **v3*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to normal vector (input) |
| *v2* | Pointer to primary color vector (input) |
| *sxy* | Pointer to address where coordinate values will be stored |
| *v3* | Pointer to color vector (output) |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation for the point *v0* is performed using a rotation matrix. Next a perspective transformation is performed and the screen coordinate *sxy* is returned.The function uses the interpolation value *p* for depth cueing, which is found by the following calculations:

LLV = LLM • v1

LC=BK + LCM • LLV

v3=p • (v2 • LC) + (1-p) • FC

v2*LC indicates the products of separate multiplications

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 15, 0) |
| *v1* -> vx, vy, vz | : (1, 3, 12) |
| *v2* -> r, g, b | : (0, 8, 0) |
| *sxy* | : (1, 15, 0), (1, 15, 0) |
| *v3* -> r, g, b | : (0, 8, 0) |
| *flag* | : (0, 32, 0) |

### Return value

1/4 of the Z component sz of the screen coordinates

### Remarks

**See also:**

## RotColorDpq3

Coordinate transformation for three points, perspective transformation, and depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotColorDpq3** (**v0*, **v1*, **v2*, **v3*, **v4*, **v5*, **v6*, **sxy0*, **sxy1*, **sxy2*, **v7*, **v8*, **v9*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**SVECTOR** **v3*, **v4*, **v5*;
**CVECTOR** **v6*;
**long** **sxy0*, **sxy1*, **sxy2*;
**CVECTOR** **v7*, **v8*, **v9*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vectors (input) |
| *v3*, *v4*, *v5* | Pointer to normal vectors (input) |
| *v6* | Pointer to primary color vector (input) |
| *sxy0*, *sxy1*, *sxy2* | Pointer to address where coordinate values will be stored |
| *v7*, *v8*, *v9* | Pointer to color vectors (output) |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of three points *v0*, *v1*, *v2* is performed using a rotation matrix. Next a perspective transformation is performed and three screen coordinates *sxy0*, *sxy1*, and *sxy2* are returned. The function uses the interpolation value *p* for depth cueing, which is found by the following calculations:

LLV0, LLV1, LLV2) = LLM • (v3, v4, v5)

(LC0, LC1, LC2) = BK + LCM • (LLV0, LLV1, LLV2)

(v7, v8, v9) = p • (v6 • (LC0, LC1, LC2)) + (1-p) • FC

Note that v6 • (LC0, LC1, LC2) indicates the products of separate multiplications.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 15, 0) |
| *v3*, *v4*, *v5* -> vx, vy, vz | : (1, 3, 12) |
| *v6* -> r, g, b | : (0, 8, 0) |
| *sxy0*, *sxy1*, *sxy2* | : (1, 15, 0), (1, 15, 0) |
| *v7*, *v8*, *v9* -> r, g, b | : (0, 8, 0) |
| *flag* | : (0, 32, 0) |

### Return value

1/4 of the Z component *sz* of the screen coordinates.

### Remarks

**See also:**

## RotColorMatDpq

Coordinate transformation, perspective transformation, and depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotColorMatDpq** (**v0*, **v1*, **v2*, **sxy*, **v3*, *matc*, *flag*)
**SVECTOR** **v0*;
**SVECTOR** **v1*;
**SVECTOR** **v2*;
**long** **sxy*;
**CVECTOR** **v3*;
**long** *matc*;
**long** *flag*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to normal vector (input) |
| *v2* | Pointer to primary color vector (input) |
| *sxy* | Pointer to address where coordinate values will be stored |
| *v3* | Pointer to color vector (output) |
| *matc* | Material (input) |
| *flag* | Address where a flag will be stored |

### Explanation

A coordinate transformation for the point *v0* is performed using a rotation matrix. Next a perspective transformation is performed and the coordinate *sxy* is returned. The function uses the interpolation value *p,* found by the following calculations, for depth cueing.

LLV = LLM • v1

LLV = LLV^ (2^matc)

LC = BK + LCM • LLV

v3 = p • (v2 • LC) + (1-p) • FC

(v2*LC) indicates separate multiplications

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 15, 0) |
| *v1* -> vx, vy, vz | : (1, 3, 12) |
| *v2* -> r, g, b | : (0, 8, 0) |
| *sxy* | : (1, 15, 0), (1, 15, 0) |
| *v3* -> r, g, b | : (0, 8, 0) |
| *matc* | : (0, 32, 0) |
| *flag* | : (0, 32, 0) |

### Return value

1/4 of the Z component sz of screen coordinates.

## Remarks

**See also:**

## RotMatrix

Finds a rotation matrix from a rotation angle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX** ***RotMatrix** (**r*, **m*)
**SVECTOR** **r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Pointer to rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue from the rotation angle (*r*[0], *r*[1], *r*[2]) in matrix *m*. A value of 4096 represents 360 degrees; and in matrices, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

*r* -> vx, vy, vz: (1, 3, 12)
(where, 1.0 stands for 360 degrees)

### Return value

This function returns *m*.

### Remarks

The matrix is obtained by doing the following multiplication. In a coordinate conversion function (such as RotTransPers) for GTE, a vector is multiplied beginning with the rightmost end. So, it is rotated around the Z-, Y-, and X-axes.

$$\begin{bmatrix} 1, & 0, & 0 \\ 0, & c0, & -s0 \\ 0, & s0, & c0 \end{bmatrix} \times \begin{bmatrix} c1, & 0 & s1 \\ 0 & 1 & 0 \\ -s1 & 0 & c1 \end{bmatrix} \times \begin{bmatrix} c2 & -s2 & 0 \\ s2 & c2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

*c0=cos (r[0]), s0=sin (r[0])

c1=cos (r[1]), s1=sin (r[1])

c2=cos (r[2]), s2=sin (r[2])

**See also:**

## RotMatrixC

Finds a rotation matrix from a rotation angle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX *RotMatrixC** (**r*, **m*)
**SVECTOR** **r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Pointer to rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

Same as RotMatrix()

### Return value

This function returns *m*.

### Remarks

This function requires a smaller table memory area than RotMatrix(), but its speed is lower.

**See also:**

## RotMatrixX

Finds a rotation matrix around the X axis.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX* RotMatrixX** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue in matrix *m* as the product of a rotation matrix around the X axis at rotation angle *r*. A value of 4096 represents a rotation angle of 360 degrees and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

m -> m[i][j] : (1, 3, 12)

r : (1, 3, 12) (360 degrees represents 1.0)

### Return value

This function returns *m*.

### Remarks

The matrix is described below.

$$\begin{bmatrix} 1, 0, 0 \\ 0, c, -s \\ 0, s, c \end{bmatrix} \times m$$

Where c = cos (r), s = sin (r)

**See also:**

## RotMatrixX_C

Finds a rotation matrix around the X axis. _C version is a smaller version.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX*** **RotMatrixX_C** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue in matrix *m* as the product of a rotation matrix around the X axis at rotation angle *r*. A value of 4096 represents a rotation angle of 360 degrees and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

m -> m[i][j] : (1, 3, 12)

r : (1, 3, 12) (360 degrees represents 1.0)

### Return value

This function returns *m*.

### Remarks

The matrix is described below.

$$\begin{bmatrix} 1, 0, 0 \\ 0, c, -s \\ 0, s, c \end{bmatrix} \times m$$

Where c = cos (r), s = sin (r)

RotMatrixX_C is a small table, low-speed C version of its RotMatrixX equivalent.

**See also:**

## RotMatrixY

Find a rotation matrix around the Y axis.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX* RotMatrixY** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle (input) |
| *m* | Pointer to rotation matrix (input/output) |

### Explanation

This function generates a rotation queue in matrix *m* as the product of a rotation matrix around the Y axis at rotation angle *r*. A value of 4096 represents a rotation angle of 360 degrees and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j] : (1, 3, 12)

*r* : (1, 3, 12) (360 degrees represents 1.0)

### Return value

This function returns *m*.

### Remarks

The matrix is described below.

$$\begin{bmatrix} c, & 0, & -s \\ 0, & 1, & 0 \\ s, & 0, & c \end{bmatrix} \times m$$

Where c = cos (r), s = sin (r)

**See also:**

## RotMatrixYXZ

Finds a rotation matrix from a rotation angle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX*** **RotMatrixYXZ** (**r*, **m*)
**SVECTOR** **r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Pointer to rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue in matrix m from the rotation angle (*r*[0], *r*[1], *r*[2]). A value of 4096 represents a rotation angle of 360 degrees, and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j] : (1, 3, 12)

*r* -> vx, vy, vz : (1, 3, 12) (360 degrees represents 1.0)

### Return value

This function returns *m*.

### Remarks

The matrix is found by performing the following multiplication. In GTE's coordinate transformation functions (such as RotTransPers()) a vector is multiplied beginning with the rightmost end. This produces rotation around the Z axis, Y axis, and X axis.

$$\begin{bmatrix} c1, 0, s1 \\ 0, 1, 0 \\ -s1, 0, c1 \end{bmatrix} \times \begin{bmatrix} 1, 0, 0 \\ 0, c0, -s0 \\ 0, s0, c0 \end{bmatrix} \times \begin{bmatrix} c2, -s2, 0 \\ s2, c2, 0 \\ 0, 0, 1 \end{bmatrix}$$

Where c0 = cos (r[0]), s0 = sin (r[0])

c1 = cos (r[1]), s1 = sin (r[1])

c2 = cos (r[2]), s2 = sin (r[2])

**See also:**

## RotMatrixYXZ_C

Finds a rotation matrix from a rotation angle. _C version is a smaller version.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX*** **RotMatrixYXZ_C** (**r*, **m*)
**SVECTOR** **r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Pointer to rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue in matrix m from the rotation angle (*r*[0], *r*[1], *r*[2]). A value of 4096 represents a rotation angle of 360 degrees, and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j] : (1, 3, 12)

*r* -> vx, vy, vz : (1, 3, 12) (360 degrees represents 1.0)

### Return value

This functions returns *m*.

### Remarks

The matrix is found by performing the following multiplication. In GTE's coordinate transformation functions (such as RotTransPers()) a vector is multiplied beginning with the rightmost end. This produces rotation around the Z axis, Y axis, and X axis.

$$\begin{bmatrix} c1,0,s1 \\ 0,1,0 \\ -s1,0,c1 \end{bmatrix} \times \begin{bmatrix} 1,0,0 \\ 0,c0,-s0 \\ 0,s0,c0 \end{bmatrix} \times \begin{bmatrix} c2,-s2,0 \\ s2,c2,0 \\ 0,0,1 \end{bmatrix}$$

Where c0 = cos (r[0]), s0 = sin (r[0])

c1 = cos (r[1]), s1 = sin (r[1])

c2 = cos (r[2]), s2 = sin (r[2])

**See also:**

## RotMatrixY_C

Find a rotation matrix around the Y axis. _C version is a smaller version.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX*** **RotMatrixY_C** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle (input) |
| *m* | Pointer to rotation matrix (input/output) |

### Explanation

This function generates a rotation queue in matrix *m* as the product of a rotation matrix around the Y axis at rotation angle *r*. A value of 4096 represents a rotation angle of 360 degrees and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j] : (1, 3, 12)

*r* : (1, 3, 12) (360 degrees represents 1.0)

### Return value

This function returns *m*.

### Remarks

The matrix is described below.

$$\begin{bmatrix} c, & 0, & -s \\ 0, & 1, & 0 \\ s, & 0, & c \end{bmatrix} \times m$$

Where c = cos (r), s = sin (r)

**See also:**

## RotMatrixZ

Finds a rotation matrix around the Z axis.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX* RotMatrixZ** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle input |
| *m* | Pointer to rotation matrix output |

### Explanation

This function generates a rotation queue in matrix m as the product of a rotation matrix around the Z axis at rotation angle *r*. A value of 4096 represents a rotation angle of 360 degrees and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j] : (1, 3, 12)

*r* : (1, 3, 12) (360 degrees represent 1.0)

### Return value

This function returns *m*.

### Remarks

The matrix is described below.

[ c, -s, 0] • m

[ s, c, 0]

[ 0, 0, 1]

$$\begin{bmatrix} c, & -s, & 0 \\ s, & c, & 0 \\ 0, & 0, & 1 \end{bmatrix} \times m$$

Where c = cos (r), s = sin (r)

**See also:**

## RotMatrixZYX_C

Find a rotation matrix around the z, y, and x axis. _C version is a smaller version.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX* RotMatrixZYX_C** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue from the rotation angle (*r*[0], *r*[1], *r*[2]) in matrix *m*. A value of 4096 represents 360 degrees; and in matrices, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

*r* -> vx, vy, vz: (1, 3, 12)
(where, 1.0 stands for 360 degrees)

### Return value

This function returns *m*.

### Remarks

The matrix is obtained by doing the following multiplication. In a coordinate conversion function (such as RotTransPers) for GTE, a vector is multiplied beginning with the rightmost end. So, it is rotated around the X axis, Y axis, and Z axis.

$$\begin{bmatrix} 1, & 0, & 0 \\ 0, & c0, & -s0 \\ 0, & s0, & c0 \end{bmatrix} \times \begin{bmatrix} c1, & 0 & s1 \\ 0 & 1 & 0 \\ -s1 & 0 & c1 \end{bmatrix} \times \begin{bmatrix} c2 & -s2 & 0 \\ s2 & c2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

*c0=cos (r[0]), s0=sin (r[0])

c1=cos (r[1]), s1=sin (r[1])

c2=cos (r[2]), s2=sin (r[2])

**See also:**

## RotMatrixZ_C

Finds a rotation matrix around the Z axis. _C is a smaller version.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX* RotMatrixZ_C** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue in matrix *m* as the product of a rotation matrix around the z axis at rotation angle *r*. A value of 4096 represents a rotation angle of 360 degrees and as a matrix element, 4096 represents 1.0.

The argument format is as follows:

m -> m[i][j] : (1, 3, 12)

r : (1, 3, 12) (360 degrees represents 1.0)

### Return value

This function returns *m*.

### Remarks

$$\text{mtx=}\begin{bmatrix} c, & -s, & 0 \\ s, & c, & 0 \\ 0, & 0, & 1 \end{bmatrix} \times m$$

RotMatrixX_C is a small table, low-speed C version of its RotMatrixZ equivalent.

**See also:**

## RotMatrix_C

Small-table, low-speed version of RotMatrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX* RotMatrix_C** (*r*, **m*)
**long** *r*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *r* | Rotation angle (input) |
| *m* | Pointer to rotation matrix (output) |

### Explanation

This function generates a rotation queue from the rotation angle (*r*[0], *r*[1], *r*[2]) in matrix *m*. A value of 4096 represents 360 degrees; and in matrices, 4096 represents 1.0.

The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

*r* -> vx, vy, vz: (1, 3, 12)
(where, 1.0 stands for 360 degrees)

### Return value

This function returns *m*.

### Remarks

The matrix is obtained by doing the following multiplication. In a coordinate conversion function (such as RotTransPers) for GTE, a vector is multiplied beginning with the rightmost end. So, it is rotated around the Z-, Y-, and X-axes.

$$\text{mtx}=\underbrace{\begin{bmatrix}1, & 0, & 0\\ 0, & c0, & -s0\\ 0, & s0, & c0\end{bmatrix}}_{\text{x-rotate}}\times\underbrace{\begin{bmatrix}c1, & 0, & s1\\ 0, & 1, & 0\\ -s1, & 0, & c1\end{bmatrix}}_{\text{y-rotate}}\times\underbrace{\begin{bmatrix}c2, & -s2, & 0\\ s2, & c2, & 0\\ 0, & 0, & 1\end{bmatrix}}_{\text{z-rotate}}$$

RotMatrixX_C is a small table, low-speed C version of its RotMatrix equivalent.

**See also:**

## RotMeshH

Performs coordinate transformation and perspective transformation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotMeshH** (**Yheight*, **Vo*, **sz*, **flag*, *Xoffset*, *Zoffset*, *m*, *n*, **base*)
**short** **Yheight*;
**DVECTOR** **Vo*;
**unsigned short** **sz*;
**unsigned short** **flag*;
**short** *Xoffset*, *Zoffset*;
**short** *m*, *n*;
**DVECTOR** **base*;

### Arguments

| | |
|---|---|
| *Yheight* | Pointer to vertex Y coordinate (input) |
| *Vo* | Pointer to screen coordinate (output) |
| *sz* | Pointer to *SZ value (output)* |
| *flag* | Pointer to flag (output) |
| *Xoffset*, *Zoffset* | Offsets for X and Z (input) |
| *m*, *n* | Number of vertices (input) |
| *base* | Pointer to base address |

### Explanation

This function performs coordinate transformation and perspective transformation for the number of quadrilateral mesh vertices indicated by m x n.

Arguments and internal data format are as follows:

| | |
|---|---|
| *Yheight* | : (1, 15, 0) |
| *Vo* -> vx, vy | : (1, 15, 0) |
| *sz* | : (0, 16, 0) |
| *flag* | : (0, 16, 0) |
| *Xoffset*, *Zoffset* | : (1, 15, 0) |
| *m*, *n* | : (1, 15, 0) |
| *base* | : (1, 15, 0) |

### Return value

None.

### Remarks

The flag must normally be set between bit 27 and bit 12 of the 32-bit flag.

**See also:**

## RotMeshPrimQ_T

Two-dimensional mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimQ_T (**
**TMERH** **msh*;
**POLY_FT4** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**SCLIP** **sclip*;
**LINE_BUF** **line_sxy*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |
| *sclip* | Pointer to screen clip area |
| *line_sxy* | Pointer to one line buffer for internal processing |

### Explanation

Perform coordinate conversion, perspective conversion, normal line clip, clipping by screen coordinates (x, y, z) and linking to OT of the following two dimensional mesh (qmesh) data.

The H direction vertex number must be a multiple of 3 (msh -> lenh=3•n).

```
1-----2-----3
|     |     |
4-----5-----6
|     |     |
7-----8-----9
```

Write texture as is (fog gathers, but do not calculate light source). Set the texture coordinates.

### Return value

None.

### Remarks

Use the following structure. The line buffer is secured above 1H + 3 vertices). If scratch pad is used as a line buffer it will be faster.

```
typedef struct {
long sminX
long smaxS
long sminY
long smaxY
long sminZ
long smaxZ
} SCLIP;
```

```
typedef struct {
long sxy
long code
} LINE_BUF;
```

**See also:**

## RotMeshPrimR_F3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_F3 (**
**TMESH** **msh*;
**POLY_F3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

With one vertex color perform flat shading (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimR_FC3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_G3 (**
**TMESH** **msh*;
**POLY_FC3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

Completely paint with one vertex color (do not calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimR_FCT3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_FCT3 (**
**TMESH** **msh*;
**POLY_FCT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

Multiply one vertex color and texture (do not calculate light source).

### Return value

None.

### Remarks

Depth cueing is not performed.

**See also:**

## RotMeshPrimR_FT3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_FT3 (**
**TMESH** **msh*;
**POLY_FT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

With one vertex color multiply the flat-shaded items and the texture (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimR_G3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_G3 (**
**TMESH** **msh*;
**POLY_G3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

With vertex color perform Gouraud shading (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimR_GC3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_GC3 (**
**TMESH** **msh*;
**POLY_GC3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

Perform Gourard complete painting with vertex color (do not calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimR_GCT3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_GCT3 (**
**TMESH** **msh*;
**POLY_GCT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

With vertex color multiply the Gouraud completely painted items and the texture (do not calculate light source).

### Return value

None.

### Remarks

Depth cueing is not performed.

**See also:**

## RotMeshPrimR_GT3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_GT3 (**
**TMESH** **msh*;
**POLY_GT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

With vertex color multiply the Gouraud-shaded items and the texture (calculate light source)

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimR_T3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimR_T3 (**
**TMESH** **msh*;
**POLY_T3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following round model mesh (rmesh) data.

```
2-----3
/ \   / \
/   \ /   \
1-----0-----4
```

With one vertex color multiply the flat-shaded items and the texture (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_F3

Round mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_F3 (**
**TMESH** **msh*;
**POLY_F3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (rmesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

With one vertex color perform flat shading (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_FC3

Strip mesh

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_FC3 (**
**TMESH** **msh*;
**POLY_FC3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

Completely paint with one vertex color (do not calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_FCT3

Strip mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_FCT3 (**
**TMESH** **msh*;
**POLY_FCT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

Multiply one vertex color and texture (do not calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_FT3

Strip mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_FT3 (**
**TMESH** **msh*;
**POLY_FT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

With one vertex color multiply the flat-shaded items and the texture (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_G3

Strip mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_G3 (**
**TMESH** **msh*;
**POLY_G3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

With vertex color perform Gouraud shading (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_GC3

Strip mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_GC3 (**
**TMESH** **msh*;
**POLY_GC3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

Perform Gouraud complete painting with vertex color (do not calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_GCT3

Strip mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_G3 (**
**TMESH** **msh*;
**POLY_GCT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

With vertex color mutliply the Gouraud completely painted items and the texture (do not calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_GT3

Strip mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotMeshPrimS_GT3 (**
**TMESH** **msh*;
**POLY_GT3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

With vertex color multiply the Gouraud-shaded items and the texture (calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotMeshPrimS_T3

Strip mesh.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotMeshPrimS_T3 (**
**TMESH** **msh*;
**POLY_T3** **prim*;
**u_long** **ot*;
**u_long** *otlen*;
**long** *dpq*;
**long** *backc*;
**)**

### Arguments

| | |
|---|---|
| *msh* | Pointer to mesh model data |
| *prim* | Pointer to GPU packet that should be created |
| *ot* | Pointer to ordering table |
| *otlen* | Ordering table length |
| *dpq* | Decides whether depth cueing will be done |
| *backc* | Decides whether back clip will be done |

### Explanation

Perform coordinate conversion, perspective conversion, and linking to *ot* of the following strip model mesh (smesh) data.

```
1-----3-----5
/ \   / \   /
/   \ /   \ /
0-----2-----4
```

Write texture as is (do not calculate light source).

### Return value

None.

### Remarks

**See also:**

## RotNclip3

Perform coordinate transformation and perspective transformation for three points, and find an interpolation value and outer product for depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotNclip3** (**v0*, **v1*, **v2*, **sxy0*, **sxy1*, **sxy2*, **p*, **otz*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**long** **sxy0*, **sxy1*, **sxy2*;
**long** **p*;
**long** **otz*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vectors (input) |
| *sxy0*, *sxy1*, *sxy2* | Pointer to address where coordinates will be stored |
| *p* | Pointer to address where an interpolation value will be stored |
| *otz* | Pointer to address where an OTZ value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of three points *v0*, *v1*, *v2* is performed using a rotation matrix. Next a perspective transformation is performed and three screen coordinates *sx0*, *sx1*, and *sx2* are returned. An interpolation value for depth cueing on *v2* to *p* is also returned. Finally, we also receive 1/4 of the Z value of the screen coordinates for *v2* to *otz*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *otz* | : (0, 32, 0) |
| *flag* | : (0, 32, 0) |

### Return value

Outer product of (*sx0, sy0*), (*sx1, sy1*), (*sx2, sy2*)

### Remarks

When the return value is negative, SX, SY, etc. are incorrect. When SX and SY are needed, use RotTransPer3().

**See also:**

## RotNclip4

Perform coordinate transformation and perspective transformation for four points, and find an interpolation value and outer product for depth cueing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotNclip4** (**v0*, **v1*, **v2*, **v3*, **sxy0*, **sxy1*, **sxy2*, **sxy3*, **p*, **otz*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**long** **sxy0*, **sxy1*, **sxy2*, **sxy3*;
**long** **p*;
**long** **otz*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vectors (input) |
| *sxy0*, *sxy1*, *sxy2*, *sxy3* | Pointer to address where coordinates will be stored |
| *p* | Pointer to address where an interpolation value will be stored |
| *otz* | Pointer to address where an OTZ value will be stored |
| *flag* | Pointer to address where a flag will be stored |

### Explanation

A coordinate transformation of four points *v0*, *v1*, *v2*, *v3* is performed using a rotation matrix. Next a perspective transformation is performed and three screen coordinates *sx0*, *sx1*, *sx2*, and *sx3* are returned. An interpolation value for depth cueing on *v2* to *p* is also returned. Finally, we also receive 1/4 of the Z value of the screen coordinates for *v2* to *otz*.

The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2*, *sxy3* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *otz* | : (0, 32, 0) |
| *flag* | : (0, 32, 0) |

### Return Value

Outer product of (sx0, sy0), (sx1, sy1), (sx2, sy2)

### Remarks

When the return value is negative, SX, SY, etc. are incorrect. When SX and SY are required, use RotTransPer4().

**See also:**

## RotPMD_F3

Independent vertex POLY_F3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_F3** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**unsigned long** **ot*;
**int** *otlen*, *id*, *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the independent vertex flat three-sided polygon-type (POLY_F3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_F4

Independent vertex POLY_F4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_F4** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the independent vertex flat four-sided polygon-type (POLY_F4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_FT3

Independent vertex POLY_FT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_FT3** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the independent vertex flat, textured three-sided polygon-type (POLY_FT3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_FT4

Independent vertex POLY_FT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_FT4** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the independent vertex flat, textured four-sided polygon-type (POLY_FT4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_G3

Independent vertex POLY_G3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_G3** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the independent vertex Gouraud-shaded three-sided polygon-type (POLY_G3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_G4

Independent vertex POLY_G4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_G4** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the independent vertex Gouraud-shaded four-sided polygon-type (POLY_G4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_GT3

Independent vertex POLY_GT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_GT3** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the independent vertex Gouraud-shaded, textured three-sided polygon-type (POLY_GT3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_GT4

Independent vertex POLY_GT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_GT4** (**pa*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the independent vertex Gouraud-shaded, textured four-sided polygon-type (POLY_GT4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_F3

Shared vertex POLY_F3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_F3** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the shared vertex flat three-sided polygon-type (POLY_F3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_F4

Shared vertex POLY_F4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_F4** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the shared vertex flat four-sided polygon-type (POLY_F4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_FT3

Shared vertex POLY_FT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_FT3** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the shared vertex flat, textured three-sided polygon-type (POLY_FT3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_FT4

Shared vertex POLY_FT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_FT4** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the shared vertex flat, textured four-sided polygon-type (POLY_FT4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_G3

Shared vertex POLY_G3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_G3** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the shared vertex Gouraud-shaded three-sided polygon-type (POLY_G3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_G4

Shared vertex POLY_G4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_G4** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the shared vertex Gouraud-shaded four-sided polygon-type (POLY_G4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_GT3

Shared vertex POLY_GT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_GT3** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all three-sided polygons included in the shared vertex Gouraud-shaded, textured three-sided polygon-type (POLY_GT3) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotPMD_SV_GT4

Shared vertex POLY_GT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotPMD_SV_GT4** (**pa*, **va*, **ot*, *otlen*, *id*, *backc*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *backc*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *backc* | Normal line clipping ON/OFF (0: ON) |

### Explanation

This function performs coordinate transformations and perspective transformations on all four-sided polygons included in the shared vertex flat four-sided polygon-type (POLY_GT4) PRIMITIVE Gp, then completes the GPU packet and links it to OT.

Only polygons with an SZ value within the range [h/2, 2^16] may be linked.

### Return value

None.

### Remarks

**See also:**

## RotRMD_F3

Independent vertex POLY_F3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_F3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_F4

Independent vertex POLY_F4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_F4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_FT3

Independent vertex POLY_FT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_FT3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_FT4

Independent vertex POLY_FT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_FT4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_G3

Independent vertex POLY_G3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_G3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_G4

Independent vertex POLY_G4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_G4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_GT3

Independent vertex POLY_GT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_GT3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_GT4

Independent vertex POLY_GT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_GT4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_F3

Shared vertex POLY_F3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_F3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_F4

Shared vertex POLY_F4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_F4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_FT3

Shared vertex POLY_FT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_FT3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_FT4

Shared vertex POLY_FT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_FT4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_G3

Shared vertex POLY_G3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_G3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_G4

Shared vertex POLY_G4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_G4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_GT3

Shared vertex POLY_GT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_GT3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotRMD_SV_GT4

Shared vertex POLY_GT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotRMD_SV_GT4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, npolygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

No polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_F3

Independent vertex POLY_F3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_F3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_F4

Independent vertex POLY_F4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_F4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_FT3

Independent vertex POLY_FT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_FT3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_FT4

Independent vertex POLY_FT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_FT4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_G3

Independent vertex POLY_G3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_G3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_G4

Independent vertex POLY_G4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_G4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_GT3

Independent vertex POLY_GT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_GT3** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_GT4

Independent vertex POLY_GT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_GT4** (**pa*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in independent vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_F3

Shared vertex POLY_F3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_F3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_F4

Shared vertex POLY_F4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_F4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_FT3

Shared vertex POLY_FT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_FT3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_FT4

Shared vertex POLY_FT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_FT4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_G3

Shared vertex POLY_G3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_G3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_G4

Shared vertex POLY_G4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_G4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_GT3

Shared vertex POLY_GT3-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_GT3** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotSMD_SV_GT4

Shared vertex POLY_GT4-type PMD data coordinate transformation and perspective transformation whose packets are then linked to the created OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotSMD_SV_GT4** (**pa*, **va*, **ot*, *otlen*, *id*, *sclip*, *hclip*, *vclip*, *nclipmode*)
**long** **pa*;
**long** **va*;
**u_long** **ot*;
**int** *otlen*;
**int** *id*;
**int** *sclip*;
**int** *hclip*;
**int** *vclip*;
**int** *nclipmode*;

### Arguments

| | |
|---|---|
| *pa* | Pointer to starting address of PRIMITIVE Gp |
| *va* | Pointer to starting address of VERTEX Gp |
| *ot* | Pointer to starting address of OT |
| *otlen* | Length of OT (number of bits) |
| *id* | Double buffer ID |
| *sclip* | Screen clip ON/OFF (ON=1) |
| *hclip* | H direction clip ([0,hclip]=display) |
| *vclip* | V direction clip ([0,vclip]=display) |
| *nclipmode* | Near Z clip mode (0=0,SCRZ/2=1) |

### Explanation

Rotates, transfers, and perspects all the polygons included in common vertex type PRIMITIVE Gp, makes GPU packets, and links them to OT.

If sclip=0, all polygons are displayed.

If sclip=1, only polygons at least one of vertices are included in the square ([0,hclip],[0,vclip]) are displayed.

If nclipmode=0, polygons are far&near clipped by sz=[0,2^16].

If nclipmode=1, polygons are far&near clipped by sz=[h,2^16] (h=distance of eye to screeen).

All polygons are backface clipped.

### Return value

None.

### Remarks

**See also:**

## RotTrans

Perform coordinate transformation using a rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void RotTrans** (**v0*, **v1*, **flag*)
**SVECTOR** **v0*;
**VECTOR** **v1*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |
| *flag* | Pointer to address where a flag is stored |

### Explanation

This function calculates *v1*=RTM • v0. The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 15, 0) |
| *v1* -> vx, vy, vz | : (1, 31, 0) |
| *flag* | : (0, 32, 0) |

### Return value

None.

### Remarks

**See also:**

## RotTransPers

Performs coordinate and perspective transformation for one vertice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotTransPers** (**vo*, **sxy*, **p*, **flag*)
**SVECTOR** **v0*;
**long** **sxy*;
**long** **p*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vertex coordinate vector (input) |
| *sxy* | Pointer to address where the screen coordinates are stored |
| *p* | Pointer to address where the interpolated value is stored |
| *flag* | Pointer to address where a flag is stored |

### Explanation

After converting the coordinate vector v0 with a rotation matrix, the function performs perspective transformation, and returns screen coordinates sx, sy. It also returns an interpolated value for depth cueing in *p*.

The argument format is as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 15, 0) |
| *sxy* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *flag* | : (0, 32, 0) |

### Return value

1/4 of the screen coordinate Z component sz.

### Remarks

**See also:**

## RotTransPers3

Perform coordinate transformation of three vertices and perspective transformation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotTransPers3** (**v0*, **v1*, **v2*, **sxy0*, **sxy1*, **sxy2*, **p*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**long** **sxy0*, **sxy1*, **sxy2*;
**long** **p*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vectors |
| *sxy0*, *sxy1*, *sxy2* | Pointer to addresses where the screen coordinates are stored |
| *p* | Pointer to address where the interpolated value is stored |
| *flag* | Pointer to address where a flag is stored |

### Explanation

After transforming the three coordinate vectors *v0*, *v1*, and *v2* using a rotation matrix, the function performs perspective transformation, and returns three screen coordinates *sxy0*, *sxy1*, and *sxy2*. It also returns to *p* an interpolated value for depth cueing corresponding to *v2*. The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *flag* | : (0, 32, 0) |

### Return value

1/4 of the screen coordinate Z component sz corresponding to *v2*.

### Remarks

**See also:**

## RotTransPers3N

Perform coordinate transformation and perspective transformation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotTransPers3N** (**v0*, **v1*, **sz*, **flag*, *n*)
**SVECTOR** **v0*;
**DVECTOR** **v1*;
**unsigned short** **sz*;
**unsigned short** **flag*;
**long** *n*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vertex coordinate vector (input) |
| *v1* | Pointer to vertex coordinate vector (output) |
| *sz* | Pointer to SZ value (output) |
| *flag* | Pointer to flag (output) |
| *n* | Number of vertices (output) |

### Explanation

This function executes the RotTransPers3() function for the number of triangles specified by n.

Arguments and internal data formats are as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 15, 0) |
| *v1* -> vx, vy | : (1, 15, 0) |
| *sz* | : (0, 16, 0) |
| *flag* | : (0, 16, 0) |

### Return value

None.

### Remarks

The flag must normally be set between bits 27 and 12 of the 32-bit flag.

**See also:**

## RotTransPers4

Perform coordinate transformation and perspective transformation for 4 vertices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long RotTransPers4** (**v0*, **v1*, **v2*, **v3*, **sxy0*, **sxyl*, **sxy2*, **sxy3*, **p*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*, **v3*;
**long** **sxy0*, **sxy1*, **sxy2*, **sxy3*;
**long** **p*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* | Pointer to vectors (input) |
| *sxy0*, *sxy1*, *sxy2*, *sxy3* | Pointer to addresses where the screen coordinates are stored |
| *p* | Pointer to address where the interpolated value is stored |
| *flag* | Pointer to address where the flag is stored |

### Explanation

After transforming the four coordinate vectors *v0*, *v1*, *v2*, and *v3* using a rotation matrix, the function performs perspective transformation, and returns four screen coordinates *sxy0*, *sxy1*, *sxy2*, and *sxy3*. It also returns an interpolated value for depth cueing to p corresponding to *v3*.The argument format is as follows:

| | |
|---|---|
| *v0*, *v1*, *v2*, *v3* -> vx, vy, vz | : (1, 15, 0) |
| *sxy0*, *sxy1*, *sxy2*, *sxy3* | : (1, 15, 0), (1, 15, 0) |
| *p* | : (0, 20, 12) |
| *flag* | : (0, 32, 0) |

### Return value

1/4 of the Z component sz of the screen coordinates corresponding to *v3*.

### Remarks

**See also:**

## RotTransPersN

Perform coordinate transformation and perspective transformation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**void RotTransPersN** (**v0*, **v1*, **sz*, **p*, **flag*, *n*)
**SVECTOR** **v0*;
**DVECTOR** **v1*;
**unsigned short** **sz*;
**unsigned short** **p*;
**unsigned short** **flag*;
**long** *n*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vertex coordinate vector (input) |
| *v1* | Pointer to vertex coordinate vector (output) |
| *sz* | Pointer to SZ value (output) |
| *p* | Pointer to intepolation value (output) |
| *flag* | Pointer to flag (output) |
| *n* | Number of vertices (output) |

### Explanation

This function performs the RotTransPers() function for the number of vertices specified by *n*.

The arguments and internal data formats are as follows:

| | |
|---|---|
| *v0* -> vx, vy, vz | : (1, 15, 0) |
| *v1* -> vx, vy | : (1, 15, 0) |
| *sz* | : (0, 16, 0) |
| *flag* | : (0, 16, 0) |

### Return value

None.

### Remarks

The flag must normally be set between bits 27 and 12 of the 32-bit flag.

**See also:**

## RotTransSV

Performs coordinate translation with rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void RotTransSV** (**SVECTOR** **v0*, **SVECTOR** **v1*, **long** **flag*)

### Arguments

| | |
|---|---|
| *v0* | Pointer to input: vector |
| *v1* | Pointer to output: vector |
| *flag* | Pointer to output: Flag |

### Explanation

RotTrans output short vector edition

v1 = RTM • v0

Argument format:

| | |
|---|---|
| *v0*>vx,vy,vy | : (1,15,0) |
| *v1*->vx,vy,vz | : (1,15,0) |
| *flag* | : (0,32,0) |

### Return value

None.

### Remarks

**See also:**

## rsin

Sine.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long rsin** (*a*)
**long int** *a*;

### Arguments

| | |
|---|---|
| *a* | Angle (in Playstation format) |

### Explanation

Finds the sine function of the angle (in Playstation format) (4096=360 degrees) using fixed-point math (where 4096=1.0).

The argument format is as follows:

a: Playstation format (4096 = 360 degrees)

Return value : (1, 19, 12)

### Return value

sin (*a*)

### Remarks

**See also:**

## ScaleMatrix

Scales a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX** ***ScaleMatrix** (**m*, **v*)
**MATRIX** **m*;
**VECTOR** **v*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (output) |
| *v* | Pointer to scale vector (input) |

### Explanation

This function scales m by v. The components of v are fixed point decimals in which 1.0 represents 4096.The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

*v* -> vx, vy, vz: (1, 3, 12)

### Return value

The function returns *m*.

### Remarks

If:

```
m =   a00,  a01,   a02
      a10,  a11,   a12
      a20,  a21,   a22
v =   [sx,     sy,     sz]
```

Then:

```
m =   a00 * sx,    a01 * sy,a02 * sz
      a10 * sx,    a11 * sy,a12 * sz
      a20 * sx,    a21 * sy,a22 * sz
```

**See also:**

## ScaleMatrixL

Scales a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX* ScaleMatrixL** (**m*, **v*)
**MATRIX** **m*;
**VECTOR** **v*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (output) |
| *v* | Pointer to scale vector (input) |

### Explanation

This function scales matrix *m* by *v*. The elements of *v* are fixed point numbers in which 4096 represents a value of 1.0.

The argument format is as follows:

m -> m[i][j] : (1, 3, 12)

v -> vx, vy, vz : (1, 3, 12)

If:

$$m = \begin{bmatrix} a00, a01, a02 \\ a10, a11, a12 \\ a20, a21, a22 \end{bmatrix}$$

$$v = [sx, sy, sz]$$

Then:

$$m = \begin{bmatrix} a00 \times sx, a01 \times sx, a02 \times sx \\ a10 \times sy, a11 \times sy, a12 \times sy \\ a20 \times sz, a21 \times sz, a22 \times sz \end{bmatrix}$$

### Return value

*m*

### Remarks

**See also:**

## SetBackColor

Sets back color vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetBackColor** (*rbk*, *gbk*, *bbk*)
**long** *rbk*, *gbk*, *bbk*;

### Arguments

*rbk*, *gbk*, *bbk*Vectors (input)

### Explanation

This function sets the back color vectors (*rbk*, *gbk*, *bbk*). Color values are in the range 0 to 255.

The argument format is as follows:

(*rbk*, *gbk*, *bbk*): (0, 32, 0)

### Return value

None.

### Remarks

**See also:**

## SetColorMatrix

Sets a local color matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetColorMatrix** (**m*)
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |

### Explanation

This function sets a local color matrix specified by m. The argument format is as follows:

*m* -> m[i][j] : (1, 3, 12)

### Return value

None.

### Remarks

**See also:**

## SetFarColor

Sets far color vectors.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetFarColor** (*rfc*, *gfc*, *bfc*)
**long** *rfc*, *gfc*, *bfc*;

### Arguments

| | |
|---|---|
| *rfc*, *gfc*, *bfc* | Vectors (input) |

### Explanation

This function sets the far color vectors (*rfc*, *gfc*, *bfc*). Color values are in the range 0 to 255.The argument format is as follows:

(*rfc*, *gfc*, *bfc*): (0, 32, 0)

### Return value

None.

### Remarks

**See also:**

## SetFogFar

Sets a fog parameter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetFogFar** (*a*, *h*)
**long** *a*, *h*;

### Arguments

| | |
|---|---|
| *a* | Z value |
| *h* | Distance |

### Explanation

When the distance between the visual point and screen is h, a defines the Z value at which the fog is 100%. A Z value which makes fog 0% is automatically set to 0.2 • *a*. *a* should satisfy 0<a<65536.

The argument format is as follows:

*a*: (0, 32, 0)

*h*: (0, 32, 0)

### Return value

None.

### Remarks

**See also:**

## SetFogNear

Sets a fog parameter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetFogNear** (*a*, *h*)
**long** *a*, *h*;

### Arguments

| | |
|---|---|
| *a* | Z value |
| *h* | Distance |

### Explanation

When the distance between the visual point and screen is h, a defines the Z value at which the fog is 0%. A Z value which makes fog 100% is automatically set to 5 • *a*. *a* should satisfy 0<a<65536 • 0.2.

The argument format is as follows:

*a*: (0, 32, 0)

*h*: (0, 32, 0)

### Return value

None.

### Remarks

**See also:**

## SetFogNearFar

Sets the fog parameters.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void SetFogNearFar** (**long** *a*, **long** *b*, **long** *h*)

### Arguments

| | |
|---|---|
| *a* | Z value with fog at 0% |
| *b* | Z value with fog at 100% |
| *h* | Distance between visual point and screen |

### Explanation

When the distance between the visual point and screen is h, the Z value with fog at 0% is set as a.

The Z value with fog at 100% is set as b.

$0 < a,b < 65536$

$(b-a) >= 100$

Argument format:

*a* : (0,32,0)

*b* : (0,32,0)

*h* : (0,32,0)

### Return value

None.

### Remarks

**See also:**

## SetGeomOffset

Sets offset values.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetGeomOffset** (*ofx*, *ofy*)
**long** *ofx*, *ofy*;

### Arguments

| | |
|---|---|
| *ofx*, *ofy* | Offset input values |

### Explanation

This function sets the offset values (*ofx*, *ofy*).

The argument format is as follows:

*ofx*, *ofy*: (1, 31, 0)

### Return value

None.

### Remarks

**See also:**

## SetGeomScreen

Sets the projection.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetGeomScreen** (*h*)
**long** *h*;

### Arguments

| | |
|---|---|
| *h* | Distance |

### Explanation

This function sets the distance h (projection) from a visual point (the eye) to the screen.

The argument format is as follows:

*h*: (0, 32, 0)

### Return value

None.

### Remarks

**See also:**

## SetLightMatrix

Sets a local light matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetLightMatrix** (**m*)
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |

### Explanation

This function sets a local light matrix specified by *m*. The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

### Return value

None.

### Remarks

**See also:**

## SetMulMatrix

Multiplies two matricies and sets one rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**MATRIX** ***SetMulMatrix** (**m0*, **m1*)
**MATRIX** **m0*, **m1*;

### Arguments

| | |
|---|---|
| *m0*, *m1* | Pointer to input matrices |

### Explanation

Multiplies two matrices and stores that value in one constant rotation matrix. The argument format is as follows:

*m0*, *m1* -> m[i][j] : (1, 3, 12)

### Return value

Returns *m0*.

### Remarks

**See also:**

## SetRGBcd

Set primary color vector and GPU code.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetRGBcd** (**v*)
**CVECTOR** **v*;

### Arguments

| | |
|---|---|
| *v* | Pointer to color vector and GPU code input: |

### Explanation

This function sets the primary color vector and GPU code *v*.

The argument format is as follows:

*v* -> r, g, b, cd : (0, 8, 0)

### Return value

None.

### Remarks

**See also:**

## SetRotMatrix

Sets a constant rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetRotMatrix** (**m*)
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |

### Explanation

This function sets a 3x3 matrix m as a constant rotation matrix. The argument format is as follows:

*m* -> m[i][j]: (1, 3, 12)

### Return value

None.

### Remarks

**See also:**

## SetTransMatrix

Setting a constant parallel transfer vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SetTransMatrix** (**m*)
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |

### Explanation

This function sets a constant parallel transfer vector specified by m. The argument format is as follows:

*m* -> t[i]: (1, 31, 0)

### Return value

None.

### Remarks

**See also:**

## Square0

Squares a vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void Square0** (**v0*, **v1*)
**VECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (L1, L2, L3) (input) |
| *v1* | Pointer to vector (L1^2, L2^2, L3^2) (output) |

### Explanation

This function returns a vector, obtained by squaring each term of the vector *v0*, to *v1*. The argument format is as follows:

*v0* -> vx, vy, vz : (1, 31, 0)

*v1* -> vx, vy, vz : (1, 31, 0)

### Return value

Returns *v1*.

### Remarks

**See also:**

## Square12

Squares a vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void Square12** (**v0*, **v1*)
**VECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (L1, L2, L3) (input) |
| *v1* | Pointer to vector (L1^2, L2^2, L3^2) (output) |

### Explanation

This function returns a vector, obtained by dividing the square of each term of the vector *v0* by 4096, to *v1*. The argument format is as follows:

*v0* -> vx, vy, vz : (1, 19, 12)

*v1* -> vx, vy, vz : (1, 19, 12)

### Return value

Returns *v1*.

### Remarks

**See also:**

## SquareRoot0

Square root.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long SquareRoot0** (*a*)
**long** *a*;

### Arguments

| | |
|---|---|
| *a* | Value |

### Explanation

This function returns the square root of a value *a*.

The argument format is as follows:

*a*: (0, 32, 0)

Return value: (0, 32, 0)

### Return value

Returns the square root of *a*.

### Remarks

**See also:**

## SquareRoot12

Square root.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long SquareRoot12** (*a*)
**long** *a*;

### Arguments

| | |
|---|---|
| *a* | Value |

### Explanation

This function returns the square root of a value *a*, which has (0, 20, 12) format, in (0, 20, 12) format.

The argument format is as follows:

*a*: (0, 20, 12)

Return value: (0, 20, 12)

### Return value

Square root of *a*.

### Remarks

**See also:**

## SubPol3

Subdivides a triangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SubPol3** (**p*, **sp*, *ndiv*)
**POL3** **p*;
**SPOL** **sp*;
**int** *ndiv*;

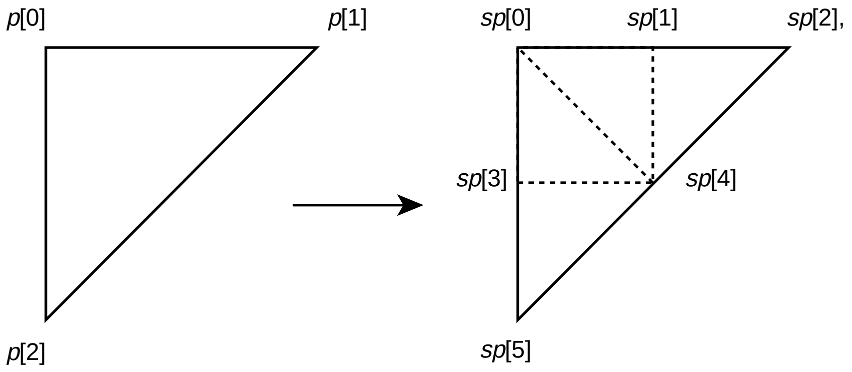
### Arguments

| | |
|---|---|
| *p* | Pointer to a 3-vertex polygon |
| *sp* | Pointer to subdivision vertex array |
| *ndiv* | Number after subdivision<br>0: None<br>1: 2x2;2: 4x4 |

### Explanation

This function subdivides a three-sided polygon *p* by the number 2**ndiv, and returns the subdivision vertex coordinates, texture coordinates, and RGB to a triangle in an array indicated by *sp*. See the figure below:

**Figure 7–2**



The argument format is as follows:

| | |
|---|---|
| *p -> sxy* | : (1, 15, 0), (1, 15, 0) |
| *p -> sz* | : (0, 16, 0) |
| *p -> uv* | : (1, 15, 0), (1, 15, 0) |
| *p -> rgb* | : (0, 8, 0), (0, 8, 0), (0, 8, 0) |
| *p -> code* | : (0, 32, 0) |
| *sp -> xy* | : (1, 15, 0), (1, 15, 0) |
| *sp -> uv* | : (1, 15, 0), (1, 15, 0) |
| *sp -> rgb* | : (0, 8, 0), (0, 8, 0), (0, 8, 0) |

### Return value

None.

## Remarks

**See also:**

## SubPol4

Subdivides a quadrangle.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void SubPol4** (**p*, **sp*, *ndiv*)
**POL4** **p*;
**SPOL** **sp*;
**int** *ndiv*;

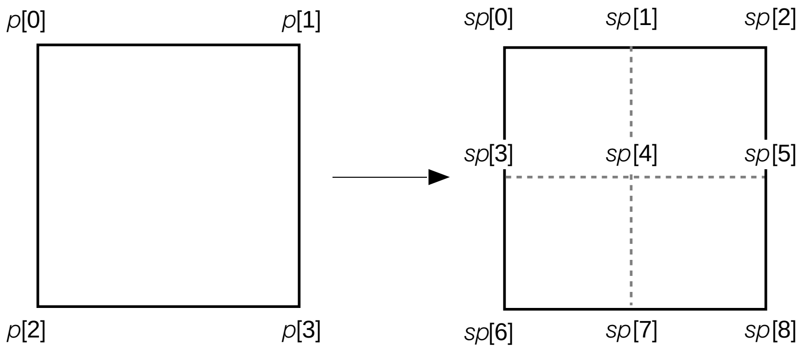
### Arguments

| | |
|---|---|
| *p* | Pointer to a 4-vertex polygon |
| *sp* | Pointer to subdivision vertex array |
| *ndiv* | Number after subdivision<br>0: None<br>1: 2x2;2: 4x4 |

### Explanation

This function subdivides a four-sided polygon p by the number 2***ndiv*, and returns the subdivision vertex coordinates, texture coordinates, and RGB to an array indicated by *sp*. See the figure below:

**Figure 7–3**



The argument format is as follows:

| | |
|---|---|
| *p* -> sxy | : (1, 15, 0), (1, 15, 0) |
| *p* -> sz | : (0, 16, 0) |
| *p* -> uv | : (1, 15, 0), (1, 15, 0) |
| *p* -> rgb | : (0, 8, 0), (0, 8, 0), (0, 8, 0) |
| *p* -> code | : (0, 32, 0) |
| *sp* -> xy | : (1, 15, 0), (1, 15, 0) |
| *sp* -> uv | : (1, 15, 0), (1, 15, 0) |
| *sp* -> rgb | : (0, 8, 0), (0, 8, 0), (0, 8, 0) |

### Return value

None.

## Remarks

**See also:**

## TransMatrix

Sets the amount of parallel transfer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX** ***TransMatrix** (**m*, **v*)
**MATRIX** **m*;
**VECTOR** **v*;

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (output) |
| *v* | Pointer to transfer vector (input) |

### Explanation

This function gives an amount of parallel transfer expressed by *v* to the matrix *m*.

The argument format is as follows:

| | |
|---|---|
| *m* -> m[i][j] | : (1, 3, 12) |
| *m* -> t[i] | : (1, 31, 0) |
| *v* -> vx, vy, vz | : (1, 31, 0) |

### Return value

This function returns m.

### Remarks

**See also:**

## TransposeMatrix

Transposes a matrix

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**MATRIX** ***TransposeMatrix** (**m0*, **m1*)
**MATRIX** **m0*, **m1*;

### Arguments

| | |
|---|---|
| *m0* | Pointer to matrix (input) |
| *m1* | Pointer to matrix (output) |

### Explanation

Transposes matrix *m0* into *m1*.

The argument format is as follows:

m0 -> m[i][j] : (1, 3, 12)

m1 -> m[i][j] : (1, 3, 12)

### Return value

Returns *m1*.

### Remarks

**See also:**

## TransRotPers

Inversely performs rotation parallel move of RotTransPers.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**long TransRotPers** (**v0*, **sxy*, **p*, **flag*)
**SVECTOR** **v0*
**long** **sxy*
**long** **p*
**long** **flag*

### Arguments

| | |
|---|---|
| *v0* | Pointer to vertex coordinate vector (input:) |
| *sxy* | Pointer to screen coordinate value (output) |
| *p* | Pointer to interpolation value (output) |
| *flag* | Pointer to flag (output) |

### Explanation

Rotates after performing a parallel move of the coordinate vector v0 with the rotation matrix.

Performs a perspective conversion and then a coordinate conversion and returns screen coordinates sx, sy.

Also, returns the interpolation value for depth cueing to p.

Argument format:

| | |
|---|---|
| *v0*->vx,vy,vz | : (1,15,0) |
| *sxy* | : (1,15,0),(1,15,0) |
| *p* | : (0,20,12) |
| *flag* | : (0,32,0) |

### Return value

1/4 of the screen coordinate Z component sz corresponding to v2.

### Remarks

**See also:**

## TransRotPers3

Inversely performs rotation parallel move of RotTransPers3.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**long TransRotPers3** (**v0*, **v1*, **v2*, **sxy0*, **sxy1*, **sxy2*, **p*, **flag*)
**SVECTOR** **v0*, **v1*, **v2*;
**long** **sxy0*, **sxy1*, **sxy2*;
**long** **p*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0*, *v1*, *v2* | Pointer to vertex coordinate vector (input) |
| *sxy0*, *sxy1*, *sxy2* | Pointer to screen coordinate value (output) |
| *p* | Pointer to interpolation value (output) |
| *flag* | Pointer to flag (output) |

### Explanation

Rotates after performing a parallel move of the three coordinate vectors v0,v1,v2 with the rotation matrix. Performs a perspective conversion and then a coordinate conversion and returns the three screen coordinates sxy0, sxy1, and sxy2.

Also, returns the interpolation value for depth cueing compatible with v2 to p.

Also, returns the screen coordinate Z item sz 1/4 compatible with v2 as the return value.

Argument format:

| | |
|---|---|
| *v0*, *v1*, *v2*->*vx*, *vy*, *vz* | : (1,15,0) |
| *sxy0*, *sxy1*, *sxy2* | : (1,15,0),(1,15,0) |
| *p* | : (0,20,12) |
| *flag* | : (0,32,0) |

### Return value

1/4 of the screen coordinate Z component sz corresponding to v2.

### Remarks

**See also:**

## TransRot_32

Inversely performs rotation parallel move of RotTrans.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**void TransRot**(**v0*, **v1*, **flag*)
**VECTOR** **v0*;
**VECTOR** **v1*;
**long** **flag*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |
| *flag* | Pointer to flag (output) |

### Explanation

After adding the 32 bit parallel move volume to v0, performs rotation with constant rotation matrix.

Argument format:

| | |
|---|---|
| *v0*->vx, vy, vz | : (1,31,0) |
| *v1*->vx, vy, vz | : (1,31,0) |
| *flag* | : (0,32,0) |

### Return value

None.

### Remarks

**See also:**

## VectorNormal

Normalize a vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**void VectorNormal** (**v0*, **v1*)
**VECTOR** **v0*;
**VECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function normalizes a vector *v0* and returns the result in *v1*.

The argument format is as follows:

v0 -> vx, vy, vz : (1, 31, 0)

v1 -> vx, vy, vz : (1, 19, 12)

### Return value

Sum of squared *v0* elements

### Remarks

**See also:**

## VectorNormalS

Normalize a vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**long VectorNormalS** (**v0*, **v1*)
**VECTOR** **v0*;
**SVECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function normalizes a vector *v0* and returns the result in *v1*.

The argument format is as follows:

v0 -> vx, vy, vz : (1, 31, 0)

v1 -> vx, vy, vz : (1, 3, 12)

### Return value

Sum of squared *v0* elements

### Remarks

The calculation will be incorrect if the sum of the squared elements of *v0* exceeds 2^31-1.

**See also:**

## VectorNormalSS

Normalize a vector.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *2.x* | *7/31/96* |

### Syntax

**long VectorNormalSS** (**v0*, **v1*)
**VECTOR** **v0*;
**SVECTOR** **v1*;

### Arguments

| | |
|---|---|
| *v0* | Pointer to vector (input) |
| *v1* | Pointer to vector (output) |

### Explanation

This function normalizes a vector *v0* and returns the result in *v1*.

The argument format is as follows:

v0 -> vx, vy, vz : (1, 16, 0)

v1 -> vx, vy, vz : (1, 3, 12)

### Return value

Sum of squared *v0* elements

### Remarks

The calculation will be incorrect if the sum of the squared elements of *v0* exceeds 2^31-1.

**See also:**

## CompMatrixLV*

Make a composite coordinate transformation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.6* | *10/23/96* |

### Syntax

**MATRIX*** **CompMatrix** (**m0*, **m1*, **m2*)
**MATRIX** **m0;* *    /* Input: Matrix */*
**MATRIX** **m1;* *    /* Input: Matrix */*
**MATRIX** **m2;* *    /* Output: Matrix */*

### Arguments

| | |
|---|---|
| *m0*, *m1* | Pointer to matrix (input) |
| *m2* | Pointer to matrix (output) |

### Explanation

This function makes a composite coordinate transformation matrix that includes parallel translation.

[m2->m] = [m0->m] * [m1->m]

(m2->t) = [m0->m] * (m1->t) + (m0->t)

Argument format

*m0* -> m[i][j] : (1, 3, 12)

*m0* -> t[i] : (1, 31, 0)

*m1* -> m[i][j] : (1, 3, 12)

*m1* -> t[i] : (1, 31, 0)

*m2* -> m[i][j] : (1, 3, 12)

*m2* -> t[i] : (1, 31, 0)

### Return value

*m2*

### Remarks

This function destroys a rotation matrix.

**See also:**

## MatrixNormal_0*

Orthonormalizes a matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.6* | *10/23/96* |

### Syntax

**void MatrixNormal_1** (**m*, **n*)
**MATRIX** **m;* */* Input: Matrix */*
**MATRIX** **n;* */* Output: Matrix */*

### Arguments

| | |
|---|---|
| *m* | Pointer to matrix (input) |
| *n* | Pointer to matrix (output) |

### Explanation

This function orthonormalizes a distorted rotation matrix.
(*m[2][0],m[2][1], and m[2][2] will be ignored.)

The argument format is as follows:

m->m[i][j]:(1.3.12)
n->m[i][j]:(1.3.12)

### Return value

None

### Remarks

**See also:**

## SetMulRotMatrix*

Multiplies constant rotation matrix by a matrix and sets one constant rotation matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.6* | *10/23/96* |

### Syntax

**MATRIX** ***SetMulRotMatrix** (**m0*;)
**MATRIX** **m0*;        */* Input: Matrix */*

### Arguments

| | |
|---|---|
| *m0* | Pointer to input matrix |

### Explanation

This function multiplies constant rotation matrix and a matrix and storestat value in one constant rotation matrix..

The argument format is as follows:

*m0* -> m[i][j] : (1, 3, 12)

### Return value

m0.

### Remarks

**See also:**

# Chapter 8: Extended Graphics Library
Table of Contents

## GsBG

BG (background surface) handler.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsBG {**
    **unsigned long** *attribute*;
    **short** *x*, *y*;
    **short** *w*, *h*;
    **short** *scrollx*, *scrolly*;
    **unsigned char** *r*, *g*, *b*;
    **GsMAP** **map*;
    **short** *mx*, *my*;
    **short** *scalex*, *scaley*;
    **long** *rotate*;
**}**;

### Members

| | |
|---|---|
| *attribute* | Attribute |
| *x*, *y* | Top left point display position |
| *w*, *h* | BG display size |
| *scrollx*, *scrolly* | x and y scroll values |
| *r*, *g*, *b* | Display brightness is set in r, g, b. (Normal brightness is 128.) |
| *map* | Pointer to map data |
| *mx*, *my* | Rotation and enlargement central point coordinates |
| *scalex*, *scaley* | Scale values in x and y directions |
| *rotate* | Rotation angle (4096 = 1 degree) |

### Explanation

For *attribute*, see the description in GsSPRITE.

BG (background) draws a large rectangle based on GsMAP data on a combination of small rectangles defined by GsCELL data. There is a GsBg for each BG. The BG may be manipulated via the GsBG structure.

To register a GsBG object in the ordering table, use GsSortBg().

*x*, *y* specifies the screen display position.

*w, h* specifies BG display size in pixels, and is not dependent on cell size or map size.

If the display area is larger than the map, the content of the map is repeatedly displayed. (Tiling function)

*scrollx*, *scrolly* specifies offset from the map display position in dots.

*r, g, b* specifies brightness values for red, green, and blue. The range is 0 to 255. 128 is the brightness of the original pattern; 255 doubles the brightness.

*map* specifies the starting address of map data with a pointer to GsMAP format map data.

*mx*, *my* specify the center of rotation and scaling as relative coordinates. The top left point of the BG is the point of origin. For example, if rotation is around the center of the BG, specify w/2 and h/2.

*scalex*, *scaley* specifies enlargement/reduction values in the x and y directions. These values are expressed in units of 4096, which stands for 1.0 (i.e. is the same size as 1.0). You can set these values up to 8 times the original size.

*rotate* specifies a rotation angle around the z-axis (4096 = 1 degree).

**Remarks**

**See also:**

## GsBOXF

Rectangle handler.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsBOXF {**
  **unsigned long** *attribute*;
  **short** *x*, *y*;
  **unsigned short** *w*, *h*;
  **unsigned char** *r*, *g*, *b*;
**}**;

### Members

| | |
|---|---|
| *attribute* | Attribute (see GsLINE attributes) |
| *x*, *y* | Display position (top left point) |
| *w*, *h* | Size of rectangle (width, height) |
| *r*, *g*, *b* | Drawing color |

### Explanation

GsBOXF is a structure used to draw a rectangle in a single color. To register GsBOXF in the ordering table, the GsSortBoxFill() function is used.

### Remarks

**See also:** GsLINE (p. 8-19).

## GsCELL

Cells constituting BG.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsCELL {**
  **unsigned short** *u*, *v*;
  **unsigned short** *cba*;
  **unsigned short** *flag*;
  **unsigned short** *tpage*;
**}**;

### Members

| | |
|---|---|
| *u* | Offset (X-direction) within the page |
| *v* | Offset (Y-direction) within the page |
| *cba* | CLUT ID |
| *flag* | An option at the time of drawing |
| *tpage* | Texture page number |

### Explanation

A rectangular array of GsCell structures is used to describe individual cells that fit together to create a BG. Each individual GsCell structure defines a rectangular portion of the overall BG.

*cba* is data that displays the position within the frame buffer of a CLUT corresponding to the cell, as follows.

**Table 8–1**

| Bit | Value |
|---|---|
| Bit 0-5 | X position of CLUT/16 |
| Bit 6-15 | Y position of CLUT |

*tpage* is a page number that indicates the position of a Sprite pattern within a frame buffer.

The *u* and *v* parameters specify the offset position for the sprite pattern within the texture page defined by *tpage*.

*flag* specifies option information for performing drawing. The meaning of each bit is as shown below.

**Table 8–2**

| Bit | Value |
|---|---|
| Bit 0 | Vertical flip (0: no flip; 1: flip) |
| Bit 1 | Horizontal flip (0: no flip; 1: flip) |

### Remarks

**See also:**

## GsCOORDINATE2

Matrix type coordinate system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsCOORDINATE2 {**
    **unsigned long** *flg*;
    **MATRIX** *coord*;
    **MATRIX** *workm*;
    **GsCOORD2PARM** **param*;
    **GsCOORDINATE2** **super*;
    **GsCOORDINATE2** **sub*;
**}**;

### Members

| | |
|---|---|
| *flg* | Flag indicating whether coord was rewritten |
| *coord* | Matrix |
| *workm* | Result of multiplication from this coordinate system to the WORLD coordinate system |
| *param* | Pointer for scale, rotation, and transfer parameters |
| *super* | Pointer to superior coordinates |
| *sub* | Not in current use |

### Explanation

GsCOORDINATE2 has superior coordinates and is defined by the matrix type *coord*.

*workm* retains the result of multiplication of matrices performed by the GsGetLw() and GsGetLs() functions in each node of GsCOORDINATE2 using the WORLD coordinates.

*flg* is referenced to omit calculations for a node for which calculations were already made, during GsGetLw() calculations. 1 means the flag is set; 0 clears the flag. The programmer must clear this flag when he has changed coord. If you neglect to clear it, the GsGetLw() and GsGetLs() functions will fail to execute normally.

*param* is used for setting coord values with layout tools.

### Remarks

*param* may be freely used if TOD animation is not used.

**See also:**

## GsDOBJ2

Used by the three-dimensional object handler GsCOORDINATE2.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Structure

**struct GsDOBJ2 {**
  **unsigned long** *attribute*;
  **GsCOORDINATE2** **coord2*;
  **unsigned long** **tmd*;
  **unsigned long** *id*;
**}**;

### Members

| | |
|---|---|
| *attribute* | Object attribute (32-bit) |
| *coord2* | Pointer to a local coordinate system |
| *tmd* | Pointer to model data |
| *id* | Reserved by the layout tool |

### Explanation

There is a GsDOBJ2 for each object of a three dimensional model; GsDOBJ2 structures may be used to manipulate the 3-dimensional model.

Use GsLinkObject4() is to link GsDOBJ2 to TMD file model data. Use GsSortObject4() to register GsDOBJ2 in the ordering table.

The *coord2* parameter is a pointer to a GsCOORDINATE2 structure defining the object's coordinate system. The location, inclination, and size of the object is defined in the matrix in this structure.

*tmd* contains the starting address of TMD model data stored in memory. *tmd* is calculated and set using GsLinkObject4().

*attribute* is 32-bit; various display attributes are set here. An explanation of each bit follows.

(a) Bits 0-2: material attenuation (not currently supported)

This sets the relationship between the normal gradient and brightness attenuation when light source calculation is performed. This takes a value of 0-3. With 0 there is no attenuation; the steepest attenuation is with 3. This parameter can be used to display an object's material quality. In general, making the attenuation steep produces a metallic quality.

Note the following points:

(1) If the material attenuation value is high, calculation takes longer and the processing requires a lot of resources.

(2) This parameter is invalid In lighting mode unless material ON is set.

(b) Bits 3-5: lighting mode

This sets the light source calculation formula. It takes a value of 0-3. The values are as listed below.

Bit 5, the highest ranking bit, is a switch to validate the lighting mode set by GsSetLightMode().

**Table 8–3: Lightning modes**

| Value | Operation |
|---|---|
| 0 | Normal mode without fog or material attenuation. This is the fastest mode and calculation takes least time. |
| 1 | Fog only mode. The fog parameter is GsFOGPARAM; set the parameter with GsSetFogParam(). |

| | |
|---|---|
| 2 | Material attenuation only mode. The amount of attenuation is set by the material attenuation bit. Not currently supported. |
| 3 | Applies both fog and material attenuation. Not currently supported. |

(c) Bit 6: Light source calculation ON/OFF switch

This bit is used when light source calculation is not performed. When light source calculation is removed, a texture-mapped polygon is displayed in the original texture color. An unmapped polygon is displayed in the model data color.

(d) Bits 7-27: Reserved, set to zero

(e) Bits 28-29: Semi-transparency rate

When semi-transparency is set to ON with bit 30, the semi-transparency rate sets the pixel-blending formula.

**Table 8–4: Semi-transparency Rate**

| Value | Processing |
|---|---|
| 0 | Normal semi-transparency processing |
| 1 | Pixel addition |
| 2 | 50% addition |
| 3 | Pixel subtraction |

(f) Bit 30: Semi-transparency ON/OFF

This sets semi-transparency ON/OFF.

This bit must be used with the uppermost bit (STP bit) of the texture color field (texture pattern when direct and CLUT color field when indexed) to set semi-transparency,. Also, the semi-transparency and non-transparency of each pixel unit may be controlled using this STP bit.

(g) Bit 31: Display ON/OFF

This turns display ON and OFF.

## Remarks

**See also:**

## GsDOBJ3

Used by the three-dimensional object handler PMD FORMAT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Structure

```
struct GsDOBJ3 {
    unsigned long attribute;
    GsCOORDINATE2 *coord2;
    unsigned long *pmd;
    unsigned long *base;
    unsigned long *sv;
    unsigned long id;
};
```

### Members

| | |
|---|---|
| *attribute* | Object attribute (32-bit) |
| *coord2* | Pointer to a local coordinate system |
| *pmd* | Pointer to model data (PMD FORMAT) |
| *base* | Pointer to object base address |
| *sv* | Pointer to shared vertex base address |
| *id* | Reserved by the layout tool |

### Explanation

There is a GsDOBJ3 for each object of a 3-dimensional model; GsDOBJ3 structures may be used to manipulate the 3-dimensional model.

Use GsLinkObject3() to link GsDOBJ3 to PMD file model data.

You can use GsDOBJ3 to access PMD data linked by GsLinkObject3(). Use GsSortObject3() to register GsDOBJ3 in the ordering table.

*coord2* is a pointer to a coordinate system unique to an object. The location, inclination, and size of the object is reflected in a matrix set in the coordinate system to point to coord2.

*pmd* retains the starting address of PMD model data stored in memory. pmd is calculated and set using GsLinkObject3().

*attribute* is 32-bit; various display attributes are set here.

Only the attribute shown below is currently available.

(a) Bits 0-30: Reserved, set to zero
(b) Bit 31: Display ON/OFF
This turns display ON and OFF.

### Remarks

*id* is not used unless the layout function is used.

**See also:**

## GsDOBJ5

Used by the three-dimensional object handler GsSortObject5.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Structure

**struct GsDOBJ5 {**
  **unsigned long** *attribute*;
  **GsCOORDINATE2** **coord2*;
  **unsigned long** **tmd*;
  **unsigned long** **packet*;
  **unsigned long** *id*;
**};**

### Members

| | |
|---|---|
| *attribute* | Object attribute (32-bit) |
| *coord2* | Pointer to local coordinate system |
| *tmd* | Pointer to model data |
| *packet* | Pointer to preset packet area |
| *id* | Reserved by the layout tool |

### Explanation

There is a GsDOBJ5 for each object of a 3-dimensional model; GsDOBJ5 structures may be used to manipulate the 3-dimensional model.

Use GsLinkObject5() to link GsDOBJ5 to TMD file model data.

You can use GsDOBJ5 to access TMD data linked by GsLinkObject5(). Use GsSortObject5() to register GsDOBJ5 in the ordering table.

*coord2* is a pointer to a coordinate system unique to an object. The location, inclination, and size of the object is reflected in a matrix set in the coordinate system to point to coord2.

*tmd* retains the starting address of TMD model data stored in memory. tmd is calculated and set using GsLinkObject5().

*packet* retains the starting address of a preset packet copied into memory. A preset packet is copied by GsPresetObject(), and is set in a GsDOBJ5 packet.

*attribute* is 32-bit; various display attributes are set here. An explanation of each bit follows.

(a) Bits 0-2: Material attenuation (not currently supported)

This sets the relationship between the normal gradient and brightness attenuation when light source calculation is performed. This takes a value of 0-3. With 0 there is no attenuation; the steepest attenuation is with 3. This parameter can be used to display an object's material quality. In general, making the attenuation steep produces a metallic quality.

Note the following points:

(1) If the material attenuation value is high, calculation takes longer and the processing requires a lot of resources.

(2) This parameter is invalid In lighting mode unless material ON is set.

(b) Bits 3-5: Lighting mode

This sets the light source calculation formula. It takes a value of 0-3. The values are as listed below.

Bit 5, the highest ranking bit, is a switch to validate the lighting mode set by GsSetLightMode().

**Table 8–5: Lightning Modes**

| Value | Operation |
|---|---|
| 0 | Normal mode without fog or material attenuation. This is the fastest mode and calculation takes least time. |
| 1 | Fog only mode. The fog parameter is GsFOGPARAM; set the parameter with GsSetFogParam(). |
| 2 | Material attenuation only mode. The amount of attenuation is set by the material attenuation bit. Not currently supported. |
| 3 | Applies both fog and material attenuation. Not currently supported. |

(c) Bit 6: Light source calculation ON/OFF switch

This bit is used when light source calculation is not performed. When light source calculation is removed, a texture-mapped polygon is displayed in the original texture color. An unmapped polygon is displayed in the model data color.

(d) Bits 7-30: Reserved, set to zero.

(e) Bits 31: Display ON/OFF

This turns display ON and OFF.

## Remarks

*id* is not used unless the layout function is used.

**See also:**

## GsFOGPARAM

Fog (depth cue) information.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsFOGPARAM {**
    **short** *dqa*;
    **long** *dqb*;
    **unsigned char** *rfc*, *gfc*, *bfc*;
**}**;

### Members

| | |
|---|---|
| *dqa* | Parameter for the degree of merging due to depth |
| *dqb* | Parameter for the degree of merging due to depth<br>For the meaning of these parameters, see the description of “FOG” in “FUNDAMENTAL GEOMETRY LIBRARY Part 1”. |
| *rfc*, *gfc*, *bfc* | Background colors |

### Explanation

*dqa* and *dqb* are background color attenuation coefficients. They can be calculated using the following formulas:

DQA = -df • 4096/64/h

DQB = 1.25 • 4096 • 4096

*df* is the distance where the attenuation coefficient is “1”; that is, the distance from the viewpoint to where the background colors are completely blended.

“h” indicates a projection, or a distance from the visual point to the screen.

### Remarks

**See also:**

## GsF_LIGHT

Parallel light source.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsF_LIGHT {**
    **long** *vx*, *vy*, *vz*;
    **unsigned char** *r*, *g*, *b*;
**}**;

### Members

| | |
|---|---|
| *vx*, *vy*, *vz* | Directional vectors for light source |
| *r*, *g*, *b* | Light colors |

### Explanation

GsF_LIGHT holds parallel light source information, and is set in the system by the GsSetFlatLight() function. Up to three parallel light sources may be set at the same time.

The light source directional vector is specified by *vx, vy, vz*. It is unnecessary for the programmer to perform normalization because the system does it. A polygon whose normal vectors are opposite to these directional vectors is exposed to the strongest light.

Light source colors are set in 8 bits by *r, g, b*.

### Remarks

**See also:**

## GsGLINE

Straight line handler with gradation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.5* | *7/31/96* |

### Structure

**struct GsLINE {**
  **unsigned long** *attribute*;
  **short** *x0*, *y0*;
  **short** *x1*, *y1*;
  **unsigned char** *r0*, *g0*, *b0*;
  **unsigned char** *r1*, *g1*, *b1*;
**}**;

### Members

| | |
|---|---|
| *attribute* | Attribute (see GsLINE attributes) |
| *x0*, *y0* | Drawing start point position |
| *x1*, *y1* | Drawing end point position |
| *r0*, *g0*, *b0* | Drawing colors of start point |
| *r1*, *g1*, *b1* | Drawing colors of end point |

### Explanation

GsGLINE is a structure used to draw straight lines with gradation. It is the same as GsLINE except that drawing colors for the starting point and end point may be specified separately.

### Remarks

**See also:**

## GsIMAGE

Information on image data composition.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

```
struct GsIMAGE {
    short pmode;
    short px, py;
    unsigned short pw, ph;
    unsigned long *pixel;
    short cx, cy;
    unsigned short cw, ch;
    unsigned long *clut;
};
```

### Members

| | |
|---|---|
| *pmode* | Pixel mode<br>0: 4-bit CLUT<br>1: 8-bit CLUT<br>2: 16-bit DIRECT<br>3: 24-bit DIRECT<br>4: Coexistence of multiple modes |
| *px*, *py* | Pixel data storage location within the frame buffer |
| *pw*, *ph* | Pixel data width and height |
| *pixel* | Pointer to pixel data |
| *cx*, *cy* | CLUT data storage location within the frame buffer |
| *cw*, *ch* | CLUT data width and height |
| *clut* | Pointer to CLUT data |

### Explanation

A structure in which TIM format data information is stored by the GsGetTimInfo() function.

### Remarks

**See also:**

## GsLINE

Straight line handler.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsLINE {**
    **unsigned long** *attribute*;
    **short** *x0*, *y0*;
    **short** *x1*, *y1*;
    **unsigned char** *r*, *g*, *b*;
**}**

### Members

| | |
|---|---|
| *attribute* | Attribute<br>Bits 28-29: Semi-transparency rate<br>0 50% x Back + 50% x Line<br>1 100% x Back + 100% x Line<br>2 100% x Back + 50% x Line<br>3 100% x Back - 100% x Line<br>Bit 30: Semi-transparency ON/OFF<br>0: Semi-transparency OFF<br>1: Semi-transparency ON<br>Bit 31<br>0: Displayed<br>1: Not displayed |
| *x0*, *y0* | Drawing start point position |
| *x1*, *y1* | Drawing end point position |
| *r*, *g*, *b* | Drawing color |

### Explanation

GsLINE is a structure for drawing straight lines. Use GsSortLine() to register a GsLINE in the ordering table.

*attribute* is 32 bits, and sets various attributes for display.

(a) Bits 0-27: Reserved, set to 0.
(b) Bits 28-29: Semi-transparency rate
If semi-transparency is turned on using bit 30, bits 28 and 29 are used to set the pixel blending method.
(c) Bit 30: Semi-transparency ON/OFF
This bit turns semi-transparency ON and OFF.
(d) Bit 31: Display ON/OFF

### Remarks

**See also:**

## GsMAP

Map comprising BG.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsMAP {**
    **unsigned char** *cellw*, *cellh*;
    **unsigned short** *ncellw*, *ncellh*;
    **GsCELL** **base*;
    **unsigned short** **index*;
**}**;

### Members

| | |
|---|---|
| *cellv*, *cellh* | Cell size (0 is treated as 256.) |
| *ncellw*, *ncellh* | Size of BG (in cells) (Not displayed if w or h is 0.) |
| *base* | Pointer to GsCELL structure array |
| *index* | Pointer to cell information |

### Explanation

GsMAP is map data used to compose BG from GsCELL. Map data are managed by cell index array information.

*cellw*, *cellh* specify the size of one cell in pixels. Note that one BG is made up of cells of the same size.

*ncellw* and *ncellh* set the size of the BG map in cells.

*base* sets the starting address of the GsCELL array.

*index* sets the starting address of the cell data table. Cell data is a list of index values whose size is equivalent to (*ncellw***ncellh*) for the array specified by *base*. If a cell value is 0xFFFF it indicates a NULL (transparent) cell.

### Remarks

**See also:**

## GsOBJTABLE2

Object table information.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsDOBJTABLE2 {**
  **GsDOBJ2** **top*;
  **int** *nobj*;
  **int** *maxnobj*;
**}**;

### Members

| | |
|---|---|
| *top* | Pointer to object array |
| *maxobj* | Size of object array |
| *nobj* | Number of valid objects in array |

### Explanation

When the three-dimensional animation function group is used, a three-dimensional object must be in the array in order to give effect to the object ID number specification. This array is called an object table. GsOBJTABLE2 contains information relating to the object table.

*top* is a pointer to the GsDOBJ2 array, within which the three-dimensional object managed by ID is created. The GsDOBJ2 array must be allocated prior to object table initialization.

*maxobj* is the size of array indicated by top; its value must be greater than the maximum value of the object handled.

*nobj* is the number of valid objects within the array.

GsOBJTABLE2 is initialized by GsInitObjTable2().

### Remarks

**See also:**

## GsOT

Ordering table header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsOT {**
**unsigned short** *length*;
**GsOT_TAG** **org*;
**unsigned short** *offset*;
**unsigned short** *point*;
**GsOT_TAG** **tag*;
**}**;

### Members

| | |
|---|---|
| *length* | Bit length of OT |
| *org* | Pointer to start address of GsOT_TAG table |
| *offset* | OT screen coordinate system Z-axis offset |
| *point* | OT screen coordinate system Z-axis typical value |
| *tag* | Pointer to GsOT_TAG currently located at the start |

### Explanation

The GsOT structure describes the header of the ordering table format supported by libgs. This header has pointers to the actual ordering table array, specified by the *org* and *tag* members. These members are initialized using th GsClearOT() function.

The *org* member always points to the start of the ordering table. The *tag* field points to the element within the ordering table at which drawing will take place.

The *length* field indicates the size of the ordering table. It is a value from 1-14 where the actual ordering table size is 2***length* (i.e. a value of 14 indicates an array of 16384 GsOT_TAG items, while a value of 8 indicates an array of 256 GsOT_TAG items).

*length* sets the size of the ordering table to values from 1-14. If the value "1" is specified, org points to a GsOT_TAG array running from 0-1. If the value "14" is specified, org points to a GsOT_TAG array running from 0-16384.

The GsClearOt() function initializes memory from *org* through to the size indicated by length. Note that memory will be destroyed if the size of the GsOT_TAG array pointed to by *org* is greater than that specified by length.

*point* is used by the GsSortOt() function in the sorting of ordering tables.

The ordering table Z-axis offset is set by *offset*. For example, if *offset* = 256, the start of the ordering table is Z = 256. (Not yet supported.)

### Remarks

*length* and *org* values should be set first. The other members are set by the GsClearOt() function.

**See also:** GsClearOt (p. 8-62), GsDrawOt (p. 8-71), GsSortOt (p. 8-152), GsCutOt(8-69).

## GsOT_TAG

Ordering table unit.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsOT_TAG {**
  **unsigned** *p : 24*;
  **unsigned char** *num : 8*;
**}**;

### Members

| | |
|---|---|
| *p* | Pointer to next item in ordering table list |
| *num* | Number of words in current GPU packet (i.e. primitive) |

### Explanation

A libgs ordering table is a linked list of GsOT_TAG structures and various types of GPU primitive structures. The *p* field of a GsOT_TAG structures indicates the least significant 24-bits of a pointer to the next item in the list. A value of 0xFFFFFF indicates the end of the list.

The GsOT structure is used by libgs to manage an array of GsOT_TAG items. Allocate an array of GsOT_TAG structures after initializing your GsOT structure.

### Remarks

**See also:**

## GsRVIEW2

Viewpoint position (Reference type).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsRVIEW2 {**
    **long** *vpx*, *vpy*, *vpz*
    **long** *vpx*, *vpy*, *vpz*
    **long** *rz*
    **GsCOORDINATE2** **super*
**}**;

### Members

| | |
|---|---|
| *vpx*, *vpy*, *vpz* | Viewpoint coordinates |
| *vrx*, *vry*, *vrz* | Reference point coordinates |
| *rz* | Viewpoint twist |
| *super* | Pointer to the coordinate system that sets the viewpoint (GsCOORDINATE2type) |

### Explanation

GsVIEW2 holds viewpoint information, and is set in libgs by the GsSetRefView2() function.

The viewpoint coordinates in the coordinate system displayed by *super* are set in *vpx, vpy, vpz*.

The reference point coordinates in the coordinate system displayed by *super* are set *in vrx, vry, vrz*.

When the z axis a vector from the viewpoint to the reference point, *rz* specifies the screen inclination against the z axis in fixed decimal format, with 4096 set to one degree.

Viewpoint and reference point coordinate systems are set in *super*. As an example of using this function, an airplane cockpit view can be realized simply by setting *super* to the airplane coordinate system.

### Remarks

**See also:**

## GsSPRITE

Sprite handler.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

**struct GsSPRITE {**
    **unsigned long** *attribute*;
    **short** *x*, *y*;
    **unsigned short** *w*, *h*;
    **unsigned short** *tpage*;
    **unsigned char** *u*, *v*;
    **short** *cx*, *cy*;
    **unsigned char** *r*, *g*, *b*;
    **short** *mx*, *my*;
    **short** *scalex*, *scaley*;
    **long** *rotate*;
**}**

### Members

| | |
|---|---|
| *attribute* | 32 bits<br>Bit 6: Brightness adjustment<br>0: OFF<br>1: ON<br>Bit 22: Vertical flip<br>0: not flipped<br>1: flipped<br>Bit 23: Horizontal flip<br>0: not flipped<br>1: flipped<br>Bits 24-25: Sprite pattern bit mode<br>0: 4-bit CLUT<br>1: 8-bit CLUT<br>2: 15-bit Direct<br>Bit 27: Rotation, enlargement, and reduction functions<br>0: ON<br>1: OFF<br>Bits 28-29: Semi-transparency rate<br>0: 50% x Back + 50% x Sprite<br>1: 100% x Back + 100% x Sprite<br>2: 100% x Back + 50% x Sprite<br>3: 100% x Back - 100% x Sprite<br>Bit 30: Semi-transparency ON/OFF<br>0: Semi-transparency OFF<br>1: Semi-transparency ON<br>Bit 31:<br>0: Displayed<br>1: Not displayed<br>NOTE: Bit 26 is not supported as yet. |
| *x*, *y* | Display position of the top left point |
| *w*, *h* | Width and height of the Sprite (Not displayed if w or h is 0.) |
| *tpage* | Sprite pattern texture page number |
| *u*, *v* | Sprite pattern offset within the page |

| | |
|---|---|
| *cx*, *cy* | Sprite CLUT address |
| *r*, *g*, *b* | Display brightness is set in r, g, b (Normal brightness is 128.) |
| *mx*, *my* | Rotation and enlargement central point coordinates |
| *scalex*, *scaley* | Scale values in x and y directions |
| *rotate* | Rotation angle (4096 = 1 degree) |

**Explanation**

GsSPRITE is a structure used to display a Sprite. This structure makes it possible to manipulate each Sprite via its parameters.

To register a GsSPRITE in the ordering table, use GsFlipSprite(), GsSortSprite(), or GsSortFastSprite().

*x, y* specifies the screen display position. (mx, my) specifies the point in the Sprite pattern used as the display position in GsSortSprite(); in GsSortFastSprite(), the point at the top left of the Sprite is used as the display position.

*w, h* specifies the width and height of the Sprite in pixels.

tpage specifies the texture page number (0-31) of the Sprite pattern.

*u, v* specifies the offset within the page from the top left point of the Sprite pattern. The range that may be specified is (0, 0) - (255, 255).

*cx, cy* specifies the starting position of CLUT (color palette) as a VRAM address. (Valid for 4-bit/8-bit mode only)

*r, g, b* specify the brightness values for red, green, and blue. The range is 0 to 255. 128 is the brightness of the original pattern; 255 doubles the brightness.

*mx, my* specify the coordinates used as the center of rotation and scaling. The top left point of the Sprite is the point of origin. For example, if rotation is around the center of the Sprite, specify w/2 and h/2.

*scalex, scaley* specifies enlargement/reduction values in the x and y directions. These values are expressed in units of 4096, which stands for 1.0 (i.e. is the same size as 1.0). You can set these values up to 8 times the original size.

*rotate* sets rotation around the z-axis according to fixed-decimal format, in which 4096 is 1 degree.

*attribute* is 32 bits, and sets various attributes for display. An explanation of each bit follows.

(a) Bits 0-5: Reserved, set to zero.

(b) Bit 6: Brightness adjustment ON/OFF switch

This bit sets Sprite pattern pixel colors according to (r, g, b) values. If this bit is set to 1, brightness is not adjusted, and (r, g, b) values are ignored.

(c) Bits 7-21: Reserved, set to zero.

(d) Bits 22-23: Vertical flipping, horizontal flipping

Sets Sprite pattern flipping display.

(e) Bits 24-25: Color mode

A Sprite pattern has 4-bit mode and 8-bit mode, both of which use the color table, and 15-bit mode, which directly displays colors. These bits are used to select any of these modes.

(f) Bit 26: Reserved, set to zero.

(g) Bit 27: Rotation enlargement/reduction function

This bit turns on or off the Sprite enlargement function. If rotation or enlargement of the Sprite is not needed, this bit should be set to OFF for high speed processing.

GsSortFastSprite() and GsSortFlipSprite() ignore this bit and always set the enlargement function to off.

(h) Bits 28-29: Semi-transparency rate

When semi-transparency is set to ON with bit 30, the semi-transparency rate sets the pixel-blending formula.

**Table 8–6: Semi-transparency Rate**

| Value | Processing |
|---|---|
| 0 | Normal semi-transparency processing |
| 1 | Pixel addition |
| 2 | 50% addition |
| 3 | Pixel subtraction |

(i) Bit 30: Semi-transparency  ON/OFF

This sets semi-transparency ON/OFF.

This bit must be used with the uppermost bit (STP bit) of the texture color field (texture pattern when direct and CLUT color field when indexed) to set semi-transparency,. Also, the semi-transparency and non-transparency of each pixel unit may be controlled using this STP bit.

(j) Bit 31: Display ON/OFF

This turns display ON and OFF.

## Remarks

**See also:**

## GsVIEW2

Viewpoint position (matrix type).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Structure

```
struct GsVIEW2 {
    MATRIX view;
    GsCOORDINATE *super
};
```

### Members

| | |
|---|---|
| *view* | Matrix used to change from superior coordinates to viewpoint coordinates |
| *super* | Pointer to the coordinate system that sets viewpoint |

### Explanation

This sets the viewpoint coordinate system. It specifies the matrix used by view to change from superior coordinates to viewpoint coordinates.

The function that sets GsVIEW2 is GsSetView2().

### Remarks

**See also:**

## _GsFCALL

The function table of GsSortObject5J(),GsSortObject4J().

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.2* | *7/31/96* |

### Structure

**struct _GsFCALL {**
 **PACKET** **(*f3[2][3])()*,**(*nf3[2])()*,**(*g3[2][3])()*,**(*ng3[2])()*;
 **PACKET** **(*tf3[2][3])()*,**(*ntf3[2])()*,**(*tg3[2][3])()*,**(*ntg3[2])()*;
 **PACKET** **(*f4[2][3])()*,**(*nf4[2])()*,**(*g4[2][3])()*,**(*ng4[2])()*;
 **PACKET** **(*tf4[2][3])()*,**(*ntf4[2])()*,**(*tg4[2][3])()*,**(*ntg4[2])()*;
**}**;

### Members

Each member is a pointer to a low-level function.

| | |
|---|---|
| *f3*, *g3*, *tf3*, *tg3*, *f4*, *g4*, *tf4*, *tg4* | Pointer to polygon types |
| First matrix: GsDivMODE_DIV/GsDivMode_NDIV | Division/no division |
| Second matrix: GsLMODE_NORMAL/GsLMODE_FOG/GsLMODE_LOFF | Light source calculation mode |
| *nf3*, *ng3*, *ntf3*, *ntg3*, *nf4*, *ng4*, *ntf4*, *ntg4* | Pointer to polygon types |
| First matrix: GsDivMODE_DIV/GsDivMode_NDIV | Division/no division |

### Explanation

GsSortObject5(),GsSortObject4() dispatches attributes, pre-set data, etc. and calls low-level functions. There are 64 low-level functions, and a single application is unlikely to use all of them.

You don't need to link GsSortObject5J() and GsSortObject4J() with unnecessary low-level functions, thereby making the code more compact. These functions are compatible with GsSortObject5() and GsSortObject4(), which organize low-level functions as tables.

_GsFCALL is the structure in which the function table is defined.The function table is organized according to polygon type, whether or not division is performed, and the light-source calculation mode.

The relevant functions are linked by entering the pointers of the appropriate low-level functions in each of the elements. It is possible to avoid linking by not including the pointers and not making extern declarations. However, if a function that does not have a pointer is called, a BUS ERROR will be generated.

The example below shows the use of GsSortObject5() with appropriate functions in all the elements. In this example, GsSortObject5J() functions the same as GsSortObject5(). This example is included in comments in the file libgs.h.

```
                                /* extern and fook only using functions */
extern _GsFCALL GsFCALL5;       /* GsSortObject5J Func Table */
jt_init()                       /* Gs SortObject5J Fook Func */
{
PACKET *GsPrstF3NL(),*GsPrstF3LFG(),*GsPrstF3L(),*GsPrstNF3();
PACKET *GsTMDdivF3NL(),*GsTMDdivF3LFG(),*GsTMDdivF3L(),*GsTMDdivNF3();
PACKET *GsPrstG3NL(),*GsPrstG3LFG(),*GsPrstG3L(),*GsPrstNG3();
PACKET *GsTMDdivG3NL(),*GsTMDdivG3LFG(),*GsTMDdivG3L(),*GsTMDdivNG3();
PACKET *GsPrstTF3NL(),*GsPrstTF3LFG(),*GsPrstTF3L(),*GsPrstTNF3();
PACKET *GsTMDdivTF3NL(),*GsTMDdivTF3LFG(),*GsTMDdivTF3L(),*GsTMDdivTNF3();
PACKET *GsPrstTG3NL(),*GsPrstTG3LFG(),*GsPrstTG3L(),*GsPrstTNG3();
PACKET *GsTMDdivTG3NL(),*GsTMDdivTG3LFG(),*GsTMDdivTG3L(),*GsTMDdivTNG3();
PACKET *GsPrstF4NL(),*GsPrstF4LFG(),*GsPrstF4L(),*GsPrstNF4();
PACKET *GsTMDdivF4NL(),*GsTMDdivF4LFG(),*GsTMDdivF4L(),*GsTMDdivNF4();
PACKET *GsPrstG4NL(),*GsPrstG4LFG(),*GsPrstG4L(),*GsPrstNG4();
PACKET *GsTMDdivG4NL(),*GsTMDdivG4LFG(),*GsTMDdivG4L(),*GsTMDdivNG4();
PACKET *GsPrstTF4NL(),*GsPrstTF4LFG(),*GsPrstTF4L(),*GsPrstTNF4();
PACKET *GsTMDdivTF4NL(),*GsTMDdivTF4LFG(),*GsTMDdivTF4L(),*GsTMDdivTNF4();
PACKET *GsPrstTG4NL(),*GsPrstTG4LFG(),*GsPrstTG4L(),*GsPrstTNG4();
PACKET *GsTMDdivTG4NL(),*GsTMDdivTG4LFG(),*GsTMDdivTG4L(),*GsTMDdivTNG4();
PACKET *GsPrstF3GNL(),*GsPrstF3GLFG(),*GsPrstF3GL();
PACKET *GsPrst3GNL(),*GsPrstF3GLFG(),*GsPrstF3GL();

  /* flat triangle */
  GsFCALL5.f3[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstF3L;
  GsFCALL5.f3[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstF3LFG;
  GsFCALL5.f3[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstF3NL;
  GsFCALL5.f3[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivF3L;
  GsFCALL5.f3[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivF3LFG;
  GsFCALL5.f3[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivF3NL;
  GsFCALL5.nf3[GsDivMODE_NDIV]                = GsPrstNF3;
  GsFCALL5.nf3[GsDivMODE_DIV]                 = GsTMDdivNF3;
  /* gour triangle */
  GsFCALL5.g3[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstG3L;
  GsFCALL5.g3[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstG3LFG;
  GsFCALL5.g3[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstG3NL;
  GsFCALL5.g3[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivG3L;
  GsFCALL5.g3[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivG3LFG;
  GsFCALL5.g3[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivG3NL;
  GsFCALL5.ng3[GsDivMODE_NDIV]                = GsPrstNG3;
  GsFCALL5.ng3[GsDivMODE_DIV]                 = GsTMDdivNG3;
  /* texture flat triangle */
  GsFCALL5.tf3[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstTF3L;
  GsFCALL5.tf3[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstTF3LFG;
  GsFCALL5.tf3[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstTF3NL;
  GsFCALL5.tf3[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivTF3L;
  GsFCALL5.tf3[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivTF3LFG;
  GsFCALL5.tf3[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivTF3NL;
  GsFCALL5.ntf3[GsDivMODE_NDIV]                = GsPrstTNF3;
  GsFCALL5.ntf3[GsDivMODE_DIV]                 = GsTMDdivTNF3;
  /* texture gour triangle */
  GsFCALL5.tg3[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstTG3L;
  GsFCALL5.tg3[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstTG3LFG;
  GsFCALL5.tg3[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstTG3NL;
  GsFCALL5.tg3[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivTG3L;
  GsFCALL5.tg3[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivTG3LFG;
  GsFCALL5.tg3[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivTG3NL;
  GsFCALL5.ntg3[GsDivMODE_NDIV]                = GsPrstTNG3;
  GsFCALL5.ntg3[GsDivMODE_DIV]                 = GsTMDdivTNG3;
  /* flat quad */
  GsFCALL5.f4[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstF4L;
  GsFCALL5.f4[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstF4LFG;
  GsFCALL5.f4[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstF4NL;
  GsFCALL5.f4[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivF4L;
  GsFCALL5.f4[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivF4LFG;
  GsFCALL5.f4[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivF4NL;
  GsFCALL5.nf4[GsDivMODE_NDIV]                = GsPrstNF4;
  GsFCALL5.nf4[GsDivMODE_DIV]                 = GsTMDdivNF4;
  /* gour quad */
  GsFCALL5.g4[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstG4L;
  GsFCALL5.g4[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstG4LFG;
  GsFCALL5.g4[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstG4NL;
  GsFCALL5.g4[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivG4L;
  GsFCALL5.g4[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivG4LFG;
  GsFCALL5.g4[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivG4NL;
  GsFCALL5.ng4[GsDivMODE_NDIV]                = GsPrstNG4;
  GsFCALL5.ng4[GsDivMODE_DIV]                 = GsTMDdivNG4;
  /* texture flat quad */
  GsFCALL5.tf4[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstTF4L;
  GsFCALL5.tf4[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstTF4LFG;
  GsFCALL5.tf4[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstTF4NL;
  GsFCALL5.tf4[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivTF4L;
  GsFCALL5.tf4[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivTF4LFG;
  GsFCALL5.tf4[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivTF4NL;
  GsFCALL5.ntf4[GsDivMODE_NDIV]                = GsPrstTNF4;
```

```
        GsFCALL5.ntf4[GsDivMODE_DIV]                = GsTMDdivTNF4;
        /* texture gour quad */
        GsFCALL5.tg4[GsDivMODE_NDIV][GsLMODE_NORMAL] = GsPrstTG4L;
        GsFCALL5.tg4[GsDivMODE_NDIV][GsLMODE_FOG]    = GsPrstTG4LFG;
        GsFCALL5.tg4[GsDivMODE_NDIV][GsLMODE_LOFF]   = GsPrstTG4NL;
        GsFCALL5.tg4[GsDivMODE_DIV][GsLMODE_NORMAL]  = GsTMDdivTG4L;
        GsFCALL5.tg4[GsDivMODE_DIV][GsLMODE_FOG]     = GsTMDdivTG4LFG;
        GsFCALL5.tg4[GsDivMODE_DIV][GsLMODE_LOFF]    = GsTMDdivTG4NL;
        GsFCALL5.ntg4[GsDivMODE_NDIV]               = GsPrstTNG4;
        GsFCALL5.ntg4[GsDivMODE_DIV]                = GsTMDdivTNG4;
        /* gradation triangle */
        GsFCALL5.f3g[GsLMODE_NORMAL]                = GsPrstF3GL;
        GsFCALL5.f3g[GsLMODE_FOG]                   = GsPrstF3GLFG;
        GsFCALL5.f3g[GsLMODE_LOFF]                  = GsPrstF3GNL;
        GsFCALL5.g3g[GsLMODE_NORMAL]                = GsPrstG3GL;
        GsFCALL5.g3g[GsLMODE_FOG]                   = GsPrstG3GLFG;
        GsFCALL5.g3g[GsLMODE_LOFF]                  = GsPrstG3GNL;
}
```

## Remarks

**See also:**

## dmyGsPrst...

Jump Table Insignificant function group (Dummy).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *dmyGsPrst... ()**

### Arguments

None

### Explanation

When this function is called for the first time, the jump table entry name is printed in standard output. It is used as an insignificant function dummy and is utilized when distinguishing which entry is being called.

### Return value

Returns the pointer to the packet.

### Remarks

For debugging use.

**See also:**

## dmyGsTMD...

Jump Table Insignificant function group (Dummy).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *dmyGsTMD... ()**

### Arguments

None

### Explanation

When this function is called for the first time, the jump table entry name is printed in standard output. It is used as an insignificant function dummy and is utilized when distinguishing which entry is being called.

### Return value

Returns the pointer to the packet.

### Remarks

For debugging use.

**See also:**

## GsA4divF3L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divF3L (TMD_P_F3** **op*, **VERT** * *vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*)

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divF3LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET** ***GsA4divF3LFG** (**TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*)

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divF3NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divF3NL (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Starting address of TMD data primitives |
| *vp* | Starting address of TMD data vertices TMD |
| *np* | Starting address of TMD data normals |
| *pk* | Top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divF4L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divF4L (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divF4LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divF4LFG (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*)

### Arguments

| | |
|---|---|
| *op* | Starting address of TMD data primitives |
| *vp* | Starting address of TMD data vertices TMD |
| *np* | Starting address of TMD data normals |
| *pk* | Top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divF4NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divF4NL** (**TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*)

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divFT3L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divFT3L (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*)

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divFT3LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divFT3LFG (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divFT3NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divFT3NL (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divFT4L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divFT4L (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divFT4LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divFT4LFG (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divFT4NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divFT4NL (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. f cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divG3L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divG3L (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divG3LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divG3LFG (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*)

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divG3NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divG3NL (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divG4L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divG4L (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divG4LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divG4LFG (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divG4NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divG4NL (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divNF3

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divNF3 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divNF4

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divNF4 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divNG3

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divNG3 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divNG4

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divNG4 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw ,ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTG3L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET * GsA4divTG3L (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTG3LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET * GsA4divTG3LFG (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTG3NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET * GsA4divTG3NL (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTG4L

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET * GsA4divTG4L (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTG4LFG

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET * GsA4divTG4LFG (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. f polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTG4NL

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divTG4NL (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTNF3

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divTNF3 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTNF4

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divTNF4 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTNG3

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divTNG3 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az ,aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsA4divTNG4

Low-level function for GsSortObject4J() (performs automatic division).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**PACKET *GsA4divTNG4 (TMD_P_F3** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch***)**

### Arguments

| | |
|---|---|
| *op* | Pointer to starting address of TMD data primitives |
| *vp* | Pointer to starting address of TMD data vertices TMD |
| *np* | Pointer to starting address of TMD data normals |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | OT shift bit |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to starting address of unused scratch pad |

### Explanation

Performs active automatic division based on Z-values, polygon size, etc.

To use these functions, they must be registered in GsFCALL4 as low-order functions, and the number of divisions must be specified in the attributes of GsDOBJ4.

Because each of these functions uses a relatively large amount of code, it would be more efficient to use only the code needed for the polygon types used.

Parameters for division include Z-values, polygon size, and GTE calculation overflow flags. These are set using the GsSetAzwh (az, aw, ah) macro.

The active division algorithm is as follows:

1. Do not divide polygons that are further away than az and that do not cause overflow in GTE calculations.
2. If cases other than 1, perform divisions (go to step 3).
3. If polygon size does not exceed aw, ah, and there is no overflow in GTE calculations, then halt division there.

Otherwise, reduce by 1/2 in the x and y directions, and divide into four sections. Call step 3 recursively. If the maximum value for divisions (the number of divisions in attribute) is reached, then halt division.

### Return value

Starting address of unused packet area.

### Remarks

GsTMDdiv functions must be registered in GsFCALL4 when using the conventional fixed division method.

**See also:** GsSetAzwh (p. 8-120).

## GsClearDispArea

Clears screen.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**GsClearDispArea** *(r*,*g*,*b)*
**unsigned char** *r*,*g*,*b*;

### Arguments

*r,g,b* Background color RGB values

### Explanation

The display area is cleared using IO.

### Return value

### Remarks

Unlike GsSortClear, a clear command is issued when GsClearDispArea() is called.

**See also:**

## GsClearOt

Initializes a libgs ordering table structure.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsClearOt** (*offset*, *point*, **otp*)
**unsigned short** *offset*;
**unsigned short** *point*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *offset* | Ordering table offset value |
| *point* | Ordering table typical value Z |
| *otp* | Pointer to ordering table |

### Explanation

This function initializes the libgs-style ordering table specified by the *otp* parameter. The *length* field of the GsOT structure must be properly set before this function is called. The *offset* parameter specifies the Z-depth value used for the start of the ordering table. The *point* offset represents the Z-depth of the entire ordering table and is used to determine depth priority when linking multiple ordering tables together.

### Return value

None.

### Remarks

**See also:**

## GsClearVcount

Clears vertical retrace counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.2* | *7/31/96* |

### Syntax

**void GsClearVcount** (*void*)

### Arguments

None.

### Explanation

This function clears the vertical retrace counter.

### Return value

None.

### Remarks

**See also:**

## GsCreateNewObj2

Creates a new object.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**GsDOBJ2 *GsCreateNewObj2** (**table*, *id*)
**GsOBJTABLE2** **table*;
**unsigned long** *id*;

### Arguments

| | |
|---|---|
| *table* | Pointer to the object table |
| *id* | ID number of the object to create |

### Explanation

Creates an object that has the ID number specified by *id* in the object table.

The superior coordinate system is WORLD and attribute is set to 0.

### Return value

Returns a pointer to the object created. NULL is returned if it fails to create the object.

### Remarks

**See also:**

## GsCutOt

OT separation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**GsOT** ***GsCutOt** (**ot_src*, **ot_dest*)
**GsOt** **ot_src*;
**GsOt** **ot_dest*;

### Arguments

| | |
|---|---|
| *ot_src* | Pointer to old OT |
| *ot_dest* | Pointer to new OT |

### Explanation

The GsCutOt() function moves the drawing commands registered in the *ot_src* ordering table to the *ot_dest* ordering table. The *length* and *tag* fields of *ot_src* are reset to zero. The *tag* field of *ot_dest* is updated to point at the drawing command which was at the start of *ot_src*. Afterwards, *ot_dest* can be used to access the ordering table.

### Return value

*ot_dest starting address.*

### Remarks

**See also:**

## GsDefDispBuff

Defines double buffers.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsDefDispBuffer** (*x0*, *y0*, *x1*, *y1*)
    **int** *x0*, *y0*;
    **int** *x1*, *y1*;

### Arguments

| | |
|---|---|
| *x0*, *y0* | Buffer 0 origin point (top left point) |
| *x1*, *y1* | Buffer 1 origin point (top left point) |

### Explanation

This function defines the display areas used for double-buffering.

The *x0* & *y0* parameters specify the coordinates within the frame buffer for buffer #0. The *x1* & *y1* parameters specify the coordinates within the frame buffer for buffer #0. Normally, buffer #0 is located at (0,0) and buffer #1 is located at (0, *yres*), where *yres* is the vertical resolution specified using the GsInitGraph() function.

If *x0*, *y0* and *x1*, *y1* are specified as the same coordinates, the double buffers are released. However, double-buffer swapping of even-numbered and odd-numbered fields is performed automatically when *x0*, *y0* and *x1*, *y1* are specified as the same coordinates in interlace mode.

The GsSwapDispBuffer() function is used to swap double buffers. The double buffer is implemented by the GPU/GTE offset. Set the libgpu or libgte offset with GsInitGraph(). When using the libgpu offset, coordinate values based on the coordinate system using the upper left point in the double buffer as the origin will be created in the packet (add the offset at the time of drawing, not at the time of packet preparation).

### Return value

None.

### Remarks

**See also:**

## GsDefDispBuff2

Defines double buffers.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsDefDispBuff 2** (*x0*, *y0*, *x1*, *y1*)
  **int** *x0*, *y0*;
  **int** *x1*, *y1*;

### Arguments

| | |
|---|---|
| *x0*, *y0* | Buffer 0 origin point (top left point) |
| *x1*, *y1* | Buffer 1 origin point (top left point) |

### Explanation

This function defines double buffer.

Differs from GsDefDispBuff only in the modification of internall variables. These modifications are not updated in libgpu and libgte until GsSwapDispBuff() is called.

Settings can be changed in the middle of the program without affecting the screen.

### Return value

None.

### Remarks

**See also:**

## GsDrawOt

Drawing for a drawing command allocated to OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsDrawOt** (**otp*)gsscale
**GsOT** **otp*;

### Arguments

*otp* Pointer to OT

### Explanation

This function starts execution of a drawing command registered in OT, specified by *otp*. Because drawing processing is performed in the background, GsDrawOt() returns immediately.

### Return value

None.

### Remarks

This function does not execute properly when GPU drawing operations are already in progress. Use ResetGraph(1) to terminate any ongoing GPU drawing operation prior to calling GsDrawOT.

**See also:**

## GsDrawOtIO

Execution drawing command (I/O version) allocated to OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.5* | *7/31/96* |

### Syntax

**void GsDrawOtIO** (**otp*)
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *otp* | Pointer to OT |

### Explanation

Starts the execution of the drawing command registered in OT, indicated by otp. Unlike GsDrawOt (), since the drawing processing is performed in the foreground, this function does not return until drawing is completed.

### Return value

None.

### Remarks

Mainly used for debugging.

**See also:**

## GsGetActiveBuffer

Gets a buffer number during drawing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**int GsGetActiveBuffer** (*void*)

### Arguments

None.

### Explanation

This function gets a double buffer index. Index values are either 0 or 1.

By entering indexes in the external variables, PSDBASEX[] and PSDBASEY[], it is possible to determine the two-dimensional address of the double buffer origin point (top left coordinates) in the frame buffer.

### Return value

Index of a double buffer (0 for buffer 0, and 1 for buffer 1)

### Remarks

**See also:**

## GsGetLs

Calculating a local screen matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsGetLs** (**coord*, **m*)
**GsCOORDINATE2** *coord;
**MATRIX** *m;

### Arguments

| | |
|---|---|
| *coord* | Pointer to local coordinates |
| *m* | Pointer to matrix |

### Explanation

This function calculates a local screen perspective transformation matrix from the GsCOORDINATE2 structure pointed to by the *coord* argument and stores the result in the MATRIX structure pointed to by the *m* argument.

For high speed operation, the function retains the result of calculation at each node of the hierarchical coordinate system. When the next GsGetLs() function is called, calculation up to the node to which no changes have been made is omitted. This is controlled by a GsCOORDINATE2 member flag (libgs replaces 1 in flags already calculated by GsCOORDINATE2).

If the contents of a superior node are changed, the effect on a subordinate node is handled by libgs, so it is not necessary to clear the flags of all subordinate nodes of the changed superior node.

### Return value

None.

### Remarks

**See also:**

## GsGetLw

Calculating a local world matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsGetLw** (**coord*, **m*)
**GsCOORDINATE2** **coord*;
**MATRIX** **m*;

### Arguments

| | |
|---|---|
| *coord* | Pointer to local coordinate system |
| *m* | Pointer to matrix |

### Explanation

This function calculates a local world perspective transformation matrix from the GsCOORDINATE2 structure pointed to by the *coord* argument and stores the result in the MATRIX structure pointed to by the *m* argument.

For high speed operation, the function retains the result of calculation at each node of the hierarchical coordinate system. When the next GsGetLw() function is called, calculation up to the node to which no changes have been made is omitted. This is controlled by a GsCOORDINATE2 member flag (libgs replaces 1 in flags already calculated by GsCOORDINATE2).

If the contents of a superior node are changed, the effect on a subordinate node is handled by libgs, so it is not necessary to clear the flags of all subordinate nodes of the changed superior node.

### Return value

None.

### Remarks

**See also:**

## GsGetLws

Calculates local world and local screen matrices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsGetLws** (**coord2*, **lw*, **ls*)
**GsCOORDINATE2** **coord2*;
**MATRIX** **lw*, **ls*;

### Arguments

| | |
|---|---|
| *coord2* | Pointer to local coordinates |
| *lw* | Pointer to matrix that stores the local world coordinates |
| *ls* | Pointer to matrix that stores the local screen coordinates |

### Explanation

GsGetLws() calculates local world and local screen coordinates. This function is faster than calling GsGetLw() followed by calling GsGetLs(). Light source calculations are performed at the time of application execution. When you use GsSetLightMatrix(), it is valid because you calculate the LW matrix.

### Return value

None.

### Remarks

**See also:**

## GsGetTimInfo

Finds TIM format header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsGetTimInfo** (**tim*, **im*)
**unsigned long** **tim*;
**GsIMAGE** **im*;

### Arguments

| | |
|---|---|
| *tim* | Pointer to TIM data |
| *im* | Pointer to an image Structure |

### Explanation

Fills in the GsIMAGE structure pointed to by the *im* parameter with the appropriate information obtained from the TIM data located at the address specified by the *tim* parameter.

### Return value

None.

### Remarks

**See also:**

## GsGetVcount

Gets the value of the vertical retrace counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**long GsGetVcount** (*void*)

### Arguments

None.

### Explanation

Obtains the value of the vertical retrace counter.

### Return value

Value of the vertical retrace counter.

### Remarks

**See also:**

## GsGetWorkBase

Gets address for storing current drawing commands.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**PACKET *GsGetWorkBase** (*void*)

### Arguments

None.

### Explanation

Allocates and returns a pointer to a buffer used for generating a drawing primitive GPU packet.

### Return value

Address to prepare the next drawing primitive packet.

### Remarks

**See also:**

## GsIncFrame

Updates the frame ID.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.2* | *7/31/96* |

### Syntax

**GsIncFrame()** (macro)

### Arguments

None.

### Explanation

GslncFrame is a macro called from within GsSwapDispBuff(). It increments the global variable PSDCNT by 1. PSDCNT is 32 bits in length, and restarts at 1 rather than 0 when it overflows.

PSDCNT is used by GsGetLw(),GsGetLs(),GsGetLws() when determining the validity of the matrix cache.

If you are not using GsSwapDispBuff() to swap double buf you must call GslncFrame to swap the buffers when you use GsGetLw(), GsGetLs(), and GsGetLws().

### Return value

None.

### Remarks

Use GsDefDispBuff() to establish settings the first time.

**See also:** GsGetLw (p. 8-77), GsGetLs (p. 8-76), GsGetLws (p. 8-78), GsSwapDispBuff (p. 8-156).

## GsInit3D

Initializes the graphics system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsInit3D**(void)

### Arguments

None.

### Explanation

GsIncFrame is a macro called from within GsSwapDispBuff().

It increments the global variable PSDCNT by 1. PSDCNT is 32 bits in length, and restarts at 1 rather than 0 when it overflows.

PSDCNT is used by GsGetLw(),GsGetLs(),GsGetLws() when determining the validity of the matrix cache.

If you are not using GsSwapDispBuff() to swap double buffers, you must call GsIncFrame to swap the buffers when you use GsGetLw(), GsGetLs(), and GsGetLws().

### Return value

None.

### Remarks

**See also:**

## GsInitCoordinate2

Initializes a local coordinate system (for use by GsCOORDINATE2).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsInitCoordinate2** (**super*, **base*)
**GsCOORDINATE2** **super*;
**GsCOORDINATE2** **base*;

### Arguments

| | |
|---|---|
| *super* | Pointer to a superior coordinate system |
| *base* | Pointer to a coordinate system (to be initialized) |

### Explanation

base->coord is indicated in the coordinate system by a single determinant, base->super is indicated with an argument, and both are initialized.

### Return value

None.

### Remarks

**See also:**

## GsInitFixBg16

High-speed BG work area initialization

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsInitFixBg16** (**bg*, **work*);
**GsBG** **bg*;
**unsigned long** **work*;

### Arguments

| | |
|---|---|
| *bg* | Pointer to GsBG |
| *work* | Pointer to work area (primitive area) |

### Explanation

This function initializes the work area used by the functions GsSortFixBg16() and GsSortFixBg32. The size of the array differs with the screen mode as follows:

size (in long units)=(((ScreenW/CellW+1)•(ScreenH/CellH+1+1)•6+4)•2+2)
ScreenH: screen height in pixels (240/480)
ScreenW: screen width in pixels (256/320/384/512/640)
CellH: cell height (in pixels)
CellW: cell width (in pixels)

Executing GsInitFixBg16()/GsInitFixBg32() once is sufficient; you need not execute it for every frame.

### Return value

None.

### Remarks

**See also:**

## GsInitFixBg32

High-speed BG work area initialization

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsInitFixBg32** (**bg*, **work*);
**GsBG** **bg*;
**unsigned long** **work*;

### Arguments

| | |
|---|---|
| *bg* | Pointer to GsBG |
| *work* | Pointer to work area (primitive area) |

### Explanation

This function initializes the work area used by the functions GsSortFixBg16() and GsSortFixBg32. The size of the array differs with the screen mode as follows:

size (in long units)=(((ScreenW/CellW+1)•(ScreenH/CellH+1+1)•6+4)•2+2)
ScreenH: screen height in dots (240/480)
ScreenW: screen width in dots (256/320/384/512/640)
CellH: cell height (in pixels)
CellW: cell width (in pixels)

Executing GsInitFixBg16()/GsInitFixBg32() once is sufficient; you need not execute it for every frame.

### Return value

None.

### Remarks

**See also:**

## GsInitGraph

Initializes the graphics system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsInitGraph** (*x_res*, *y_res*, *int1*, *dither*, *vram*)
  **int** *x_res*;
  **int** *y_res*;
  **int** *int1*;
  **int** *dither*;
  **int** *vram*;

### Arguments

| | |
|---|---|
| *x_res* | Horizontal resolution (256/320/384/512/640) |
| *y_res* | Vertical resolution (240/480) |
| *intl* | Interlace display flag (bit 0)<br>0: Non-interlace GsNONINTR<br>1: Interlace GsINTER<br>Double buffer offset mode (bit 2)<br>0: GTE offset GsOFSGTE<br>3: GPU offset GsOFSGPU |
| *dither* | Dithering processing flag<br>0: OFF<br>1: ON |
| *vram* | VRAM mode<br>0: 16-bit<br>1: 24-bit |

### Explanation

This function resets “gpu”, and initializes the libgs graphics system.

libgpu settings recognize the global variables GsDISPENV and GsDRAWENV, so the programmer can reference libgpu settings and changes.

*x_res* specificies horizontal resolution (256/320/384/512/640), *y_res* vertical resolution, and bit 0 of *int1* sets interlace/non-interlace display. A vertical 480-line interlace is only effective when used in conjunction with a VGA monitor. Note that even when interlace is 240 lines, the top and bottom eight lines cannot usually be seen on domestic televisions.

The default offset mode of bit2 of *int1* is determined by whether the double-buffer offset is a GTE or GPU offset. Since the double buffer offset values in the packet realized by the GPU are not added, this is the easier to handle alternative.

In 24bit mode only memory image display is possible, and no polygons can be drawn using the GPU.

Graphics system initialization includes GsIDMATRIX and GsIDMATRIX2 initialization, so Gs library functions do not perform normally after GsInitGraph() is called until these items are reset by your program.

### Return value

None.

### Remarks

**See also:**

## GsInitGraph2

Initializes the graphics system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsInitGraph2** (*x_res*, *y_res*, *int1*, *dither*, *vram*)
  **int** *x_res*;
  **int** *y_res*;
  **int** *int1*;
  **int** *dither*;
  **int** *vram*;

### Arguments

| | |
|---|---|
| *x_res* | Horizontal resolution (256/320/384/512/640) |
| *y_res* | Vertical resolution (240/480) |
| *intl* | Interlace display flag (bit 0) |
| | 0: Non-interlace |
| | 2: Interlace |
| | Double buffer offset mode (bit 2) |
| | 0: GTE offset |
| | 2: GPU offset |
| *dither* | Dither ON/OFF during drawing |
| | 0: OFF |
| | 1: ON |
| *vram* | VRAM mode |
| | 0: 16-bit |
| | 1: 24-bit |

### Explanation

GsInitGraph2 is different from GsInitGraph in that the GPU is not initialized COLD. This function is useful for changing libgs resolution without affecting screen synchronization.

### Return value

None.

### Remarks

Always use GsInitGraph() for the first initialization.

**See also:**

## GsInitObjTable2

Initializes the object table.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsInitObjTable2** (**obj_tbl*, **obj_area*, **obj_coord*, **obj_cparam*, *nobj*)
**GsOBJTABLE2** **obj_tbl*;
**GsDOBJ2** **obj_area*;
**GsCOORDINATE2** **obj_coord*;
**GsCOORD2PARAM** **obj_cparam*;
**long** *nobj*;

### Arguments

| | |
|---|---|
| *obj_tbl* | Pointer to an object table |
| *obj_area* | Pointer to a GsDOBJ2 array |
| *obj_coord* | Pointer to a GsCOORDINATE2 array |
| *obj_cparam* | Pointer to a GsCOORD2PARAM array |
| *nobj* | Maximum object number (size of array) |

### Explanation

Carries out initialization of the object table displayed by *obj_tbl* and also carries out initialization of three-dimensional objects within the array of GsDOBJ2, which is indicated by *obj_area*. The following parameters are set for initialized objects.

| | |
|---|---|
| ID number | GsOBJ_UNDEF ( = 0xFFFFFFFF) |
| Parent object | WORLD ( = 0) |
| TMD address | 0 |
| Coordinate system | A factor of same order in the array which is indicated by *obj_coord* |

Because each of the objects managed by ID has an object system, it prepares the arrays of GsCOORDINATE2 and GsCOORD2PARAM which are the same size as *obj_tbl*, and the initialization is carried out in such a way that each same order factor responds.

### Return value

None.

### Remarks

**See also:**

## GsInitVcount

Initializes vertical retrace counter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsInitVcount**(void)

### Arguments

None.

### Explanation

This function initializes the vertical retrace counter, and starts it.

### Return value

None.

### Remarks

**See also:**

## GsLinkObject3

Links an object with PMD data (For GsSortObject3).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsLinkObject3** (**pmd*, **obj_base*)
**unsigned long** **pmd*;
**GsDOBJ3** **obj_base*;

### Arguments

| | |
|---|---|
| *pmd* | Pointer to starting address of the PMD data to be linked |
| *obj_base* | Pointer to array of the object structure to be linked |

### Explanation

Links GsDOBJ3 object structure to all objects contained in the PMD data, so that the PMD format three-dimensional object modelled can be handled by GsDOBJ3.

### Return value

None

### Remarks

Unlike GsLinkObject4(), it is not possible to select and link an object included in the PMD data. All objects contained in *pmd* will be linked to the object handler array beginning with *obj_base*.

**See also:**

## GsLinkObject4

Links an object to TMD data (For GsSortObject4).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsLinkObject4** (*tmd*, **obj_base*, *n*)
**unsigned long** *tmd*;
**GsDOBJ2** **obj_base*;
**unsigned long** *n*;

### Arguments

| | |
|---|---|
| *tmd* | Starting address of the TMD data to be linked |
| *obj_base* | Array of the object structure to be linked |
| *n* | Index of the object to be linked |

### Explanation

Links GsDOBJ2 object structure to the *n*-th object of the TMD data so that the TMD format three-dimensional object modelled can be handled by GsDOBJ2.

### Return value

None

### Remarks

An object linked using GsLinkObject4() uses GsSortObject4() to create a packet.

**See also:**

## GsLinkObject5

Links an object to TMD data (For GsSortObject5).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsLinkObject5** (*tmd*, **obj_base*, *n*)
**unsigned long** *tmd*;
**GsDOBJ5** **obj_base*;
**unsigned long** *n*;

### Arguments

| | |
|---|---|
| *tmd* | Starting address of the TMD data to be linked |
| *obj_base* | Array of the object structure to be linked |
| *n* | Index of the object to be linked |

### Explanation

Links GsDOBJ5 object structure to the *n*-th object of the TMD data so that the TMD format three-dimensional object modelled can be handled by GsDOBJ5.

### Return value

None

### Remarks

**See also:**

## GsMapModelingData

Maps TMD data to real addresses.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsMapModelingData** (**p*)
**unsigned long** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to starting address of TMD data |

### Explanation

TMD data includes various fields which contain the memory addresses of certain pieces of data. However, during the preparation of TMD data, the memory address where the data will be loaded is not yet known. Therefore, address fields in the TMD data are stored as offsets from the start of the data, The GsMapModelingData function changes these offsets into actual addresses after the TMD data has been loaded into memory. This must be done before the TMD data may be used.

### Return value

None.

### Remarks

A flag is set in the TMD data whose offset addresses have been converted into real addresses. So, no side effect occurs even if GsMapModelingData() is called again.

**See also:**

## GsMulCoord0

MATRIX multiplication.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsMulCoord2** (**m1*, **m2*)
**MATRIX** **m1*, **m2*

### Arguments

| | |
|---|---|
| *m1*, *m2* | Pointer to matrix |

### Explanation

This function multiplies MATRIX *m2* by the translation matrix. The results are stored in *m3*.

*m3* = *m1* x *m2*

### Return value

None.

### Remarks

**See also:**

## GsMulCoord2

MATRIX multiplication.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsMulCoord2** (**m1*, **m2*)
**MATRIX** **m1*, **m2*

### Arguments

| | |
|---|---|
| *m1*, *m2* | Pointer to matrix |

### Explanation

GsMulCoord2 multiplies the MATRIX *m2* by the translation matrix *m1*and stores the result in *m2*.

*m2* = *m1* x *m2*

### Return value

None.

### Remarks

**See also:**

## GsMulCoord3

MATRIX multiplication.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsMulCoord3** (**m1*, **m2*)
**MATRIX** **m1*, **m2*

### Arguments

*m1*, *m2* Pointer to matrix

### Explanation

GsMulCoord3 multiplies the MATRIX *m2* by the translation matrix *m1*and stores the result in *m2*.

*m1* = *m1* x *m2*

### Return value

None.

### Remarks

**See also:**

## GsPresetObject

Creates a preset packet for a GsDOBJ5-type object.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long** ***GsPresetObject** (**objp*, **addr*)
**GsDOBJ5** **objp*;
**unsigned long** **addr*;

### Arguments

| | |
|---|---|
| *objp* | Pointer to the object to be preset |
| *addr* | Pointer to starting address of the area in which the preset packet is to be prepared. |

### Explanation

Presetting refers to the advance preparation of polygons of all objects as packets. The areas that need not be rewritten (e.g., U and V of texture) for each frame will not be rewritten, thus ensuring high speed.

The return value of GsPresetObject points to the address next to the last preset address, so when presetting the next object, preserve the return value and pass it as an argument of the next GsPresetObject(). The return value will indicate how large an area must be allocated for the preset area.

A GsDOBJ5 type object pointer is exclusively used for presetting.

### Return value

Pointer that indicates the next to the last preset address.

### Remarks

**See also:**

## GsPrstF3GL

Flat-shaded triangle (light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *GsPrstF3GL** (**TMD_P_F3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, *i***nt** *n*, *i***nt** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

This is a low-level function of GsSortObject5J() and must be registered with GsFCALL5.

This function performs coordinate & perspective transformation, backface clipping, and light source calculation for *n* triangles, creates the GPU packet in the buffer, and links it into the OT. There must be two preset packets in the buffer per polygon.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsPrstF3GLFG

Flat-shaded triangle (light source calculation + FOG).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *GsPrstF3GLFG** (**TMD_P_F3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

This is a low-level function of GsSortObject5J() and must be registered with GsFCALL5.

This function performs coordinate & perspective transformation, backface clipping, and light source calculation for *n* triangles, creates the GPU packet in the buffer, and links it into the OT. There must be two preset packets in the buffer per polygon.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsPrstF3GNL

Flat-shaded triangle (without light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *GsPrstF3GNL** (**TMD_P_F3G** **op*,**VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, *i***nt** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

This is a low-level function of GsSortObject5J() and must be registered with GsFCALL5.

This function performs coordinate & perspective transformation, backface clipping, and light source calculation for *n* triangles, creates the GPU packet in the buffer, and links it into the OT. There must be two preset packets in the buffer per polygon.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsPrstF3L, GsPrstF3LFG, GsPrstF3NL, GsPrstNF3

TMD data flat triangle processing.

GsPrstF3L: flat triangle (light source calculation)

GsPrstF3LFG: flat triangle (light source calculation + FOG)

GsPrstF3NL: flat triangle (without light source calculation)

GsPrstNF3: flat triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstF3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstF3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstF3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstNF3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_F3** **primtop*;
(**TMD_P_NF3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_F3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstNF3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsPrstF3L(), Gs PrstF3LFG()), for n (number of) flat triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer. (see libgs: PresetObject)

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsPrstF4L, GsPrstF4LFG, GsPrstF4NL, GsPrstNF4

TMD data flat quadrilateral processing

GsPrstF4L: flat rectangle (light source calculation)

GsPrstF4LFG: flat rectangle (light source calculation + FOG)

GsPrstF4NL: flat rectangle (without light source calculation)

GsPrstNF4: flat rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstF4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstF4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstF4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstNF4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_F4** **primtop*;
(**TMD_P_NF4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_F4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstNF4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsPrstF4L(), Gs PrstF4LFG()), for n (number of) flat rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer (see libgs: GsPresetObject).

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsPrstG3GL

Gouraud-shaded triangle (light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**PACKET *GsPrstG3GL** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, *i***nt** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

This is a low-level function of GsSortObject5J() and must be registered with GsFCALL5.

This function performs coordinate & perspective transformation, backface clipping, and light source calculation for *n* triangles, creates the GPU packet in the buffer, and links it into the OT. There must be two preset packets in the buffer per polygon.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsPrstG3GLFG

Gouraud-shaded triangle (light source calculation + FOG).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**PACKET *GsPrstG3GLFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

This is a low-level function of GsSortObject5J() and must be registered with GsFCALL5.

This function performs coordinate & perspective transformation, backface clipping, and light source calculation for *n* triangles, creates the GPU packet in the buffer, and links it into the OT. There must be two preset packets in the buffer per polygon.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsPrstG3GNL

Gouraud-shaded triangle (without light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**PACKET *GsPrstG3GNL** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, *i***nt** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

This is a low-level function of GsSortObject5J() and must be registered with GsFCALL5.

This function performs coordinate & perspective transformation, backface clipping, and light source calculation for *n* triangles, creates the GPU packet in the buffer, and links it into the OT. There must be two preset packets in the buffer per polygon.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsPrstG3L, GsPrstG3LFG, GsPrstG3NL, GsPrstNG3

TMD data Gouraud-shaded, triangle processing.

GsPrstG3L: Gouraud triangle (light source calculation)

GsPrstG3LFG: Gouraud triangle (light source calculation + FOG)

GsPrstG3NL: Gouraud triangle (without light source calculation)

GsPrstNG3: Gouraud triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstG3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstG3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstG3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstNG3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_G3** **primtop*;
(**TMD_P_NG3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_G3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstNG3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsPrstG3L(), Gs PrstG3LFG()), for n (number of) Gouraud triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer.

(see libgs: GsPresetObject)

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsPrstG4L, GsPrstG4LFG, GsPrstG4NL, GsPrstNG4

TMD data Gouraud-shaded, quadrilateral processing.

GsPrstG4L: Gouraud rectangle (light source calculation)

GsPrstG4LFG: Gouraud rectangle (light source calculation + FOG)

GsPrstG4NL: Gouraud rectangle (without light source calculation)

GsPrstNG4: Gouraud rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstG4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstG4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstG4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstNG4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_G4** **primtop*;
(**TMD_P_NG4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_G4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstNG4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsPrstG4L(), Gs PrstG4LFG()), for n (number of) Gouraud rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer. (see libgs: GsPresetObject)

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsPrstTF3L, GsPrstTF3LFG, GsPrstTF3NL, GsPrstTNF3

TMD data flat, textured triangle processing.

GsPrstTF3L: flat textured triangle (light source calculation)

GsPrstTF3LFG: flat textured triangle (light source calculation + FOG)

GsPrstTF3NL: flat textured triangle (without light source calculation)

GsPrstTNF3: flat textured triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstTF3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTF3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTF3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTNF3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TF3** **primtop*;
(**TMD_P_TNF3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_FT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstTNF3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsPrstTF3L(), Gs PrstTF3LFG()), for n (number of) flat textured triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer (see libgs: GsPresetObject).

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsPrstTF4L, GsPrstTF4LFG, GsPrstTF4NL, GsPrstTNF4

TMD data flat, textured quadrilateral processing.

GsPrstTF4L: flat textured rectangle (light source calculation)

GsPrstTF4LFG: flat textured rectangle (light source calculation + FOG)

GsPrstTF4NL: flat textured rectangle (without light source calculation)

GsPrstTNF4: flat textured rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstTF4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTF4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTF4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTNF4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TF4** **primtop*;
(**TMD_P_TNF4** **primtop*;
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_FT4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstTNF4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation GsPrstTF4L(), Gs PrstTF4LFG()), for n (number of) flat textured rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer (see libgs: GsPresetObject).

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsPrstTG3L, GsPrstTG3LFG, GsPrstTG3NL, GsPrstTNG3

TMD data Gouraud-shaded, textured triangle processing.

GsPrstTG3L: Gouraud texture triangle (light source calculation)

GsPrstTG3LFG: Gouraud texture triangle (light source calculation + FOG)

GsPrstTG3NL: Gouraud texture triangle (without light source calculation)

GsPrstTNG3: Gouraud texture triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstTG3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTG3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTG3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTNG3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TG3** **primtop*;
(**TMD_P_TNG3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstTNG3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsPrstTG3L(), Gs PrstTG3LFG()), for n (number of) Gouraud texture triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer (see libgs: GsPresetObject).

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsPrstTG4L, GsPrstTG4LFG, GsPrstTG4NL, GsPrstTNG4

TMD data Gouraud-shaded, textured quadrilateral processing.

GsPrstTG4L: Gouraud texture rectangle (light source calculation)

GsPrstTG4LFG: Gouraud texture rectangle (light source calculation + FOG)

GsPrstTG4NL: Gouraud texture rectangle (without light source calculation)

GsPrstTNG4: Gouraud texture rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsPrstTG4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTG4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTG4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsPrstTNG4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TG4** **primtop*;
(**TMD_P_TNG4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsPrstTNG4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsPrstTG4L(), Gs PrstTG4LFG()), for n (number of) Gouraud texture rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

Two packets per polygon are preset in the buffer (see libgs: GsPresetObject).

See libgs in the Run-time Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsRemoveObj2

Deletes an object.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**GsDOBJ2** ***GsRemoveObj2** (**table*, *id*)
**GsOBJTABLE2** **table*;
**unsigned long** *id*;

### Arguments

| | |
|---|---|
| *table* | Pointer to the object table |
| *id* | ID number of the object to delete |

### Explanation

GsRemoveObj2() searches for an object with the ID number specified by the object table, and returns it. The value of the vacant area ID is set in GsOBJ_UNDEF without filling the vacant area that has occurred.

### Return value

Returns pointer to object upon successful deletion. Returns NULL for failure.

### Remarks

**See also:**

## GsScaleScreen

Scales the screen coordinate system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.2* | *7/31/96* |

### Syntax

**void GsScaleScreen (**
**SVECTOR** **scale*
**)**

### Arguments

| | |
|---|---|
| *scale* | Pointer to the scale factor (12 bit fixed radix point format) Always set the factor in relation to the original screen coordinate systems set on GsSetView2 () and GsSetRefView2 (). When ONE is inserted into vx, vy or vz it returns to the original. |

### Explanation

GsScaleScreen ( ) scales the screen coordinate system against the world coordinates.

Unlike the world coordinates which have 32 bits of space, the screen coordinates have only 16 bits. Accordingly, this brings about problems such as FarClip being close.

In order to solve this, GsScaleScreen is provided to scale the screen coordinates and cover a larger area than world.

For example, when specifying ONE/2 to vx, vy or vz, the screen coordinate system is expanded to the equivalent of 17 bits. However, since the precision is 16 bits, the lower 1 bit will be invalid.

Attention must be paid here to make sure that the screen coordinate system which has a different scale is not registered to the OT with the same scale.

For example, in order to register an object calculated with the normal scaling screen coordinate system to the OT which has already registered an object with a 1/2 screen coordinate system scale, it is necessary to shift the excess 1 bit before registering.

When the scaling matrix set by GsScaleScreen to the external variable GsWSMATRIX, and the screen coordinates set by GsSetView2 and GsSetRevView2 to the external variable GsWSMATRIX_ORG are defined, the WSMATRIX is held.

### Return value

None.

### Remarks

**See also:**

## GsSearchObjByID2

Object search.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**GsDOBJ2 *GsSearchObjByID2** (**table*, *id*);
**GsOBJTABLE2** **table*;
**unsigned long** *id*;

### Arguments

| | |
|---|---|
| *table* | Pointer to the object table |
| *id* | ID number of the object to be found |

### Explanation

Finds the object specified by *id* within the object table specified by *table*.

### Return value

Returns a pointer to the relevant object, or NULL if object not found.

### Remarks

**See also:**

## GsSearchTmdByID

Searches for modeling data within TMD data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long** ***GsSearchTmdByID** (**tmd*, **id_list*, *id*);
**unsigned long** **tmd*;
**int** **id_list*;
**int** *id*;

### Arguments

| | |
|---|---|
| *tmd* | Pointer to TMD data |
| *id_list* | Pointer to the model data ID list |
| *id* | ID number of the model data |

### Explanation

Searches within the TMD data specified by *tmd* for the model data specified by *id*.

### Return value

This function returns a pointer to the requested model data. This value may be entered instead of the value of the *tmd* field of the GsDOBJ2 structure. NULL is returned if the requested model data cannot be found within the buffer specified by *tmd*.

### Remarks

**See also:**

## GsSetAmbient

Set color and brightness of ambient lighting.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetAmbient** (*r*, *g*, *b*)
**unsigned short** *r*, *g*, *b*;

### Arguments

| | |
|---|---|
| *r*, *g*, *b* | Ambient color RGB values (0-65535) |

### Explanation

This function sets the color and brightness of the ambient lighting in the 3D world. Values for red, green, and blue are set independently. A value of 4096 corresponds to normal ambient brightness, 0 to minimum brightness. Values greater than 4096 strengthen that color.

### Return value

None.

### Remarks

**See also:**

## GsSetAzwh

Sets conditions for active subdivision.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**void GsSetAzwh (**
**int z,**
**short w,**
**short h**
**)**

### Arguments

| | |
|---|---|
| *z* | Critical near z value for activate subdivision |
| *w*, *h* | Size of polygon within subdivision routine at which no more subdivision will be done |

### Explanation

Sets the conditions for active subdivision.

Z is the near z value for the start of the subdivision fragmentation and *w*, *h* is the polygon size for halting the subdivision.

### Return value

None.

### Remarks

**See also:**

## GsSetClip

Sets a drawing clipping area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSetClip** (**clip*)
**RECT** **clip*;

### Arguments

| | |
|---|---|
| *clip* | Beginning address of a RECT structure for setting a clipping area |

### Explanation

Sets clipping for drawing. This function is different from GsSetDrawBuffClip() in that its argument can be used to specify a clip area. Note that this clipping value is a relative one within the double buffer, and thus the clip position will not change even if the double buffer is swapped.

### Return value

None

### Remarks

Clipping is done by libgpu.

**See also:**

## GsSetClip2

Sets a drawing clipping area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**DRAWENV *GsSetClip2**(**clip*)
**RECT** **clip*;

### Arguments

*clip* Beginning address of a RECT structure for setting a clipping area

### Explanation

Sets the clipping rectangle for drawing to the rectangle specified by *clip*. This function is different from GsSetClip in that the DRAWENV and DISPENV structures are not updated. The return value of GsSetClip2 is a pointer to a DRAWENV structure that can be used if necessary to set the system DRAWENV structure using PutDrawEnv. Note that the global DRAWENV must have been previously specified in order for the information in this structure to be valid.

Note that this clipping rectangle is relative to whichever is the current buffer, even if double-buffering is used.

### Return value

Returns a pointer to an updated DRAWENV structure (which can be used to update the system DRAWENV structure if desired).

### Remarks

Clipping is done by libgpu.

**See also:**

## GsSetClip2D

Sets two-dimensional clipping.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**GsSetClip2D** (**rectp*)
**RECT** **rectp*;

### Arguments

| | |
|---|---|
| *rectp* | Pointer to the area to be clipped. |

### Explanation

This function sets the area given by RECT as the area to be clipped.

This setting is affected by the double buffer. This means that the function leads to the automatic clipping of the same area even though the double buffer has been swapped.

GsSetDrawBuffClip must be invoked in order to validate this setting immediately afterwards.

If GsSetDrawBuffClip is not specifically invoked, the setting is valid from the next frame.

### Return value

None.

### Remarks

**See also:**

## GsSetDrawBuffClip

Sets drawing clipping area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetDrawBuffClip** (*void*)

### Arguments

None.

### Explanation

This function sets clipping for drawing. The clipping value set by GsClip2D() is set in libgs.

This clipping value is a relative one within the double buffers. The clipping position does not change when double buffers are swapped.

### Return value

None.

### Remarks

This function does not execute correctly if GPU drawing is in progress. Use ResetGraph(1) to terminate any current drawing process or DrawSync() to wait until the process is completed.

**See also:**

## GsSetDrawBuffOffset

Sets the drawing offset.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetDrawBuffOffset** (*void*)

### Arguments

None.

### Explanation

GsSetDrawBuffOffset sets the drawing offset. The offset value set in the global variable "POSITION" is updated.

This offset is relative within the double buffer. The offset value is preserved even if double buffers are swapped.

GsSetDrawBuffOffset sets the libgte or libgpu offset.

**Note:**

Using the GsOFSGPU or GsOFSGTE macro for the third argument of GsInitGraph() determines whether the libgte or libgpu offset should be set.

### Return value

None.

### Remarks

This function does not execute correctly if GPU drawing is in progress. Use ResetGraph(1) to terminate any current drawing process or DrawSync() to wait until the process is completed.

**See also:**

## GsSetFlatLight

Sets parallel light source.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetFlatLight** (*id*, **lt*)
**unsigned int** *id*;
**GsF_LIGHT** **lt*;

### Arguments

| | |
|---|---|
| *id* | Light source number (0, 1, 2) |
| *lt* | Pointer to light source data |

### Explanation

GsSetFlatLight sets a parallel light source. Up to three light sources (ID = 0, 1, 2) may be set. Light source data is given GsF_LIGHT structure.

### Return value

None.

### Remarks

Note that even when the contents of the GsF_LIGHT structure are written over, the setting will be not reflected in libgs unless this function is invoked.

**See also:**

## GsSetFogParam

Sets the fog parameter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetFogParam** (**fogparam*)
**GsFOGPARAM** **fogparam*;

### Arguments

| | |
|---|---|
| *fogparam* | Pointer to a fog parameter structure |

### Explanation

GsSetFogParam sets the fog parameter. Fog is valid only in lighting mode 1 and 3. (Light mode 3 is not supported.)

### Return value

None.

### Remarks

**See also:**

## GsSetLightMatrix

Sets a light matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetLightMatrix** (**mp*)
**MATRIX** **mp*;

### Arguments

| | |
|---|---|
| *mp* | Pointer to matrix |

### Explanation

This function multiplies the local screen light matrix *mp* by the matrices for the three light vectors, and places the results in libgte.

When using libgte during application execution of light source calculations, GsSetLightMatrix() must be set in advance.

Depending on the type of model data, some GsSortObject...() will calculate the light source during execution. In this case, also, you must use GsSetLightMatrix() to set a light matrix in advance.

Matrices to be set as GsSetLightMatrix() arguments are usually local screen matrices.

### Return value

None.

### Remarks

**See also:**

## GsSetLightMatrix2

Sets a light matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSetLightMatrix** (**mp*)
**MATRIX** **mp*;

### Arguments

*mp* Pointer to matrix

### Explanation

The three light source vector matrices and the local screen light matrix *mp*, passed as a parameter, are multiplied and placed in libgte.

This matrix must be set in advance when performing light-source calculations using libgte. GsSortObject... may perform light-source calculations during execution, depending on the type of modeling data handled. You must use GsSetLightMatrix() to set the light matrix in these cases as well.

Generally, the matrix set as a parameter in GsSetLightMatrix() will be a local world matrix.

### Return value

None.

### Remarks

The difference between GsSetLightMatrix() and this function is whether the GTE rotation matrix and the parameter mp are destroyed or not. GsSetLightMatrix2() destroys these values, however, GsSetLightMatrix2() is faster than GsSetLightMatrix().

You must call GsSetLightMatrix() before GsSetLsMatrix().

**See also:**

## GsSetLightMode

Sets light source mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetLightMode** (*mode*)
**int** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Light source mode value (0-3)<br>0: Normal lighting<br>1: Normal lighting with fog ON<br>2: Material lighting (not currently supported)<br>3: Material lighting with fog ON (not currently supported) |

### Explanation

This function sets the default light source mode. The method of light source calculation can be also set using status bits for each object. The setting of the status bit overrides the default setting.

### Return value

None.

### Remarks

**See also:**

## GsSetLsMatrix

Sets a local screen matrix.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetLsMatrix** (**mp*)
**MATRIX** **mp*;

### Arguments

| | |
|---|---|
| *mp* | Pointer to local screen matrix to be set |

### Explanation

This function sets a GTE local screen matrix. When you use GsSetLsMatrix for LIBGTE perspective transform processing, you must first set a local screen matrix in LIBGTE.

For GsSortObject---() calls to perform perspective transformations and use them in LIBGTE, you must first execute GsSetLsMatrix.

### Return value

None.

### Remarks

**See also:**

## GsSetOffset

Sets an offset.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSetOffset** (*offx*, *offy*)
**int** *offx*;
**int** *offy*;

### Arguments

| | |
|---|---|
| *offx* | Drawing offset X |
| *offy* | Drawing offset Y |

### Explanation

Specifies a drawing offset. This function is different from GsSetDrawBuffOffset() in that it sets an offset provided as an argument while GsSetDrawBuffOffset() sets a value for the global variable, POSITION. The offset to be provided as an argument is a relative offset inside the double buffer. In other words, the double buffer base offset is added to the offset provided by the argument.

Using the GsOFSGPU or GsOFSGTE macro for the third argument of GsInitGraph() determines whether the libgte or libgpu offset should be set.

### Return value

None.

### Remarks

This function does not execute correctly if GPU drawing is in progress. Use ResetGraph(1) to terminate any current drawing process of DrawSync() to wait until the process is completed.

**See also:**

## GsSetOrign

The offset is valid if the screen is switched.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.2* | *7/31/96* |

### Syntax

**void GsSetOrign (int** *x* **, int** *y***)**

### Arguments

| | |
|---|---|
| *x* | Drawing offset X |
| *y* | Drawing offset Y |

### Explanation

Specifies a drawing offset. The function is different to GsSetOffset( ) in that the offset value set in GsSetOffset() is temporary and becomes invalid when GsSwapDispBuff ( ) and GsSetDrawBuffOffset () are called, while the offset value set in GsSetOrign () is valid until the GsSetOrign( )is called again.

The offset to be provided as an argument is a relative offset inside the double buffer. In other words, the double buffer base offset is added to the offset provided by the argument.

The location is the same as the GsSetClip2D( ) in clipping.

Note: Using the GsOFSGPU or GsOFSGTE macro for the third argument of GsInitGraph() determines whether the libgte or libgpu offset should be set.

In fact they are set by the external variable POSITIONs offx, offy.

### Return value

None.

### Remarks

This function does not execute correctly when GPU drawing is in progress, so it is necessary to call this function after terminating drawing using ResetGraph (1).

**See also:**

## GsSetProjection

Sets the projection plane position.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetProjection** (*h*)
**unsigned long** *h*;

### Arguments

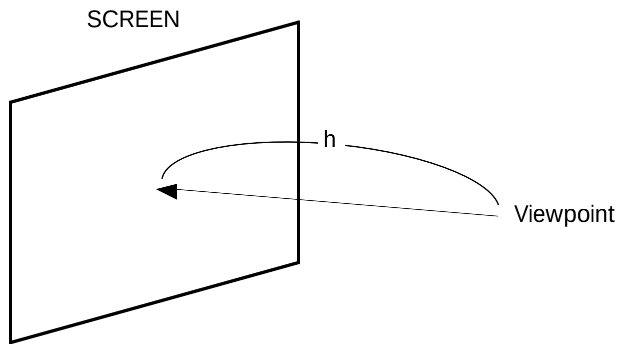
| | |
|---|---|
| *h* | Distance (projection) between the viewpoint and projection plane<br>Default: 1000 |

### Explanation

This function adjusts the drawing angle.

A projection is the distance from the viewpoint to the projection plane.

**Figure 8–1: Projection**



The size of the projection plane is specified by (xres, yres) in the GsInitGraph() function. The size of the projection plane is constant with respect to the resolution, so the drawing angle is reduced as projection is increased, and the drawing angle is increased as projection is decreased.

Depending on the resolution, the aspect ratio may not be 1 to 1. In this case, set the X coordinate scale to 1/2 and adjust the aspect ratio.

**Table 8–7: Resolution and Aspect Ration**

| Resolution | Aspect ration |
|---|---|
| 640x480 | 1:1 |
| 640x240 | 2:1 |
| 320x240 | 1:1 |

### Return value

None.

### Remarks

**See also:**

## GsSetRefView2

Sets viewpoint position.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**int GsSetRefView2** (**pv*)
**GsRVIEW2** **pv*;

### Arguments

| | |
|---|---|
| *pv* | Pointer to viewpoint position information |

### Explanation

Calculates WSMATRIX using viewpoint information. *pv* is a pointer to a GsRVIEW2 structure.

Since WSMATRIX will not change unless the viewpoint is moved, it need not be called for each frame. However, if the viewpoint is moved, WSMATRIX must be called for each frame in order for changes to be updated.

Call GsSetRefView2() for each frame if the GsRVIEW2 member super is set to anything other than WORLD; even if the other parameters are not changed, if the parameters of the superior coordinate system are changed, the viewpoint will have moved.

### Return value

Upon success, the function returns 0. Upon failure, it returns 1.

### Remarks

**See also:**

## GsSetRefView2L

Sets viewpoint (High Precision Version).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.5* | *7/31/96* |

### Syntax

**int GsSetRefView2L** (**pv*)
**GsRVIEW2** **pv*;

### Arguments

*pv* Pointer to viewpoint location information (view/reference point type)

### Explanation

This function calculates WSMATRIX using the viewpoint information. The parameter is structure GsRVIEW2. It is not necessary to call this function for every frame if the viewpoint is not changed since WSMATRIX is not changed. However, if the viewpoint changes, this must be called for every frame to update.

When setting a GsRVIEW2 member, "super" to values other than WORLD, GsSetRefFiew2L() must be called for every frame since the viewpoint moves in result of parent coordinate parameter change even when other parameters are not changed. The difference between GsSetRefView2L() and GsSetRefView2() is precision level. In GsSetRefView2L(), viewpoint whobbling caused by insufficient precision is improved compared with GsSetRefView2().

The execution time of GsSetRefView2L(),however, is doubled.

### Return value

0 for successful viewpoint set

1 for error.

### Remarks

**See also:**

## GsSetTodFrame2

Manages TOD data of the frame section.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long** **GsSetTodFrame2* (*fn*, **dp*, **table*, **tmd_id*, **tmd*, *mode*)
**int** *fn*;
**unsigned** **dp*;
**GsOBJTABLE2** **table*;
**int** **tmd_id*;
**unsigned** **tmd*;
**int** *mode*;

### Arguments

| | |
|---|---|
| *fn* | Current frame number |
| *dp* | Pointer to TOD data |
| *table* | Pointer to object table |
| *tmd_id* | Pointer to TMD ID list |
| *tmd* | Pointer to TMD data |
| *mode* | Gives the class of packet to be executed:<br>GsTOD_CREATE<br>Executes entire packet<br>GsTOD_NOCREATE<br>Does not carry out creation/deletion of object<br>GsTOD_COORDONLY<br>Executes coordinate change only |

### Explanation

Renews the object's parameters according to the content of the packet group of one frame within TOD data. The tmd_id and tmd value are not referenced when the mode value is made GsTOD_COORDONLY.

### Return value

It returns the pointer to the TOD data after execution. This value always indicates the start of the TOD data of one frame section.

### Remarks

**See also:**

## GsSetTodPacket2

Manages TOD data of one packet section.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long** ***GsSetTodPacket2** (**dp*, **tbl*, **tmd_id*, **tmd*, *mode*)
**unsigned** **dp*
**GsOBJTABLE2** **tbl*;
**int** **tmd_id*;
**unsigned** **tmd*;
**int** *mode*;

### Arguments

| | |
|---|---|
| *dp* | Pointer to TOD data that is executing |
| *tbl* | Pointer to object table |
| *tmd_id* | Pointer to model data list |
| *tmd* | Pointer to TMD data |
| *mode* | Gives the class of the packet to be executed:<br>GsTOD_CREATE<br>Executes the entire packet<br>GsTOD_NOCREATE<br>Does not carry out object creation/deletion<br>GsTOD_COORDONLY<br>Executes coordinate change only |

### Explanation

Manages the data of 1 packet section from TOD data.

### Return value

It returns a pointer to TOD data after execution.

### Remarks

**See also:**

## GsSetView2

Sets viewpoint.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**int GsSetView2***(*pv)*
**GsVIEW2** **pv*;

### Arguments

| | |
|---|---|
| *pv* | Pointer to viewpoint position data (matrix form) |

### Explanation

Sets the WS matrix directly.

If you use GsSetRefView2() to determine the WS matrix from the viewpoint and the focal point, insufficient precision may cause errors when you move the viewpoint; it is more effective to use GsSetView2().

When the GsVIEW2 "super" member is set to anything besides WORLD, even if the other parameters remain unchanged, the viewpoint will move if the parent coordinate system parameters are changed. In such cases, you must call GsSetRefView2() for each frame.

If GsIDMATRIX2 is used as the base matrix, then the aspect ratio of the screen will be adjusted automatically.

### Return value

Settings successful: 0; 1 if unsuccessful.

### Remarks

**See also:**

## GsSetWorkBase

Sets address for storing drawing commands.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSetWorkBase** (**base_addr*)
**PACKET** **base_addr*;

### Arguments

*base_addr* Pointer to an address storing drawing commands

### Explanation

This function sets the memory address for storing drawing primitives generated by functions like GsSortObject...(), GsSortSprite(), and GsSortBg().

Primitives must be stored at the starting address of a packet area reserved by the user at the beginning of processing for each frame.

### Return value

None.

### Remarks

**See also:**

## GsSortBg, GsSortFastBg

Registers BG in the OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortBg** (**bg*, **otp*, *pri*)
**GsBG** **bg*;
**GsOT** **otp*;
**unsigned short** *pri*;

**void GsSortFastBg** (**bg*, **otp*, *pri*)
**GsBG** **bg*;
**GsOT** **otp*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *bg* | Pointer to BG |
| *otp* | Pointer to OT |
| *pri* | Position in OT |

### Explanation

This function assigns BG indicated by *bg* to the ordering table indicated by *otp*. *pri* refers to the priority of the Sprite in the ordering table. The highest priority is zero, with the lowest priority depending on the size of the ordering table. Values beyond the ordering table size are clipped to the available maximum value.

Turning off extension and rotation functions in the bg attributes gives higher-speed processing.

In GsSortFastBg(), not using enlargement, rotation, and reduction functions results in higher-speed processing. The Sprite structure members values *mx*, *my*, *scalex*, *scaley*, and *rotate* are ignored.

### Return value

None.

### Remarks

**See also:**

## GsSortBoxFill

Registers rectangle in the OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSortBoxFill** (**bp*, **ot*, *pri*)
**GsBOXF** **bp*;
**GsOT** **ot*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *bp* | Pointer to GsBOXF |
| *ot* | Pointer to OT |
| *pri* | Position in OT |

### Explanation

This function assigns a rectangle indicated by *bp* to the ordering table indicated by *ot*.

### Return value

None.

### Remarks

**See also:**

## GsSortClear

Registers a screen clear command in the OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSortObject** (*r*, *g*, *b*, **otp*)
**unsigned char** *r*, *g*, *b*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *r*, *g*, *b* | Background color RGB values |
| *otp* | Pointer to OT |

### Explanation

Sets a screen clear command at the start of the OT indicated by *otp*.

### Return value

None.

### Remarks

This function only registers a screen clear command in the OT; actual clearing will not be executed until the GsDrawOt() function is used to start drawing.

This function is called after GsSwapDispBuff().

**See also:**

## GsSortFixBg16

Registers high-speed BG in the OT

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortFixBg16** (**bg*, **work*, **otp*, *pri*);
**GsBG** **bg*;
**unsigned long** **work*;
**GsOT** **otp*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *bg* | Pointer to GsBG |
| *work* | Pointer to work area (primitive area) |
| *otp* | Pointer to OT |
| *pri* | Position in OT |

### Explanation

This function performs high-speed BG registration processing. It is less CPU-intensive than GsSortFastBg(), with the following restrictions.

- BG rotation/enlargement/reduction is not possible
- Fixed cell size: 16 for GsSortFixBg16, 32 for GsSortFixBg32
- Texture patter color mode is only 4-bit/8-bit
- Map size is optional
- Scroll is possible (in 1-pixel units)
- Only full-screen

This function uses the work area to store drawing primitives. The work area uses an unsigned long array; this must be initialized beforehand by GsInitFixBg16() or GsInitFixBg32(). This function does not use the packet area (an area set by GsSetWorkBase()).

### Return value

None.

### Remarks

**See also:**

## GsSortFixBg32

Registers high-speed BG in the OT

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortFixBg32** (**bg*, **work*, **otp*, *pri*);
**GsBG** **bg*;
**unsigned long** **work*;
**GsOT** **otp*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *bg* | Pointer to GsBG |
| *work* | Pointer to work area (primitive area) |
| *otp* | Pointer to OT |
| *pri* | Position in OT |

### Explanation

This function performs high-speed BG registration processing. It is less CPU-intensive than GsSortFastBg(), with the following restrictions.

- BG rotation/enlargement/reduction is not possible
- Fixed cell size: 16 for GsSortFixBg16, 32 for GsSortFixBg32
- Texture patter color mode is only 4-bit/8-bit
- Map size is optional
- Scroll is possible (in 1-pixel units)
- Only full-screen

This function uses the work area to store drawing primitives. The work area uses an unsigned long array; this must be initialized beforehand by GsInitFixBg16() or GsInitFixBg32(). This function does not use the packet area (an area set by GsSetWorkBase()).

### Return value

### Remarks

**See also:**

## GsSortGLine

Registers straight line in the OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortGLine** (**lp*, **ot*, *pri*)
**GsLINE** **lp*;
**GsOT** **ot*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *lp* | Pointer to GsLINE/GsGLINE |
| *ot* | Pointer to OT |
| *pri* | Position in OT |

### Explanation

This function assigns the straight line indicated by *lp* to the ordering table indicated by *ot*.

The GsSortLine() function registers single-color straight lines in OT, and the GsSortGLine() function graded straight lines in OT.

### Return value

None.

### Remarks

**See also:**

## GsSortLine

Registers straight line in the OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortLine** (**lp*, **ot*, *pri*)
**GsLINE** **lp*;
**GsOT** **ot*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *lp* | Pointer to GsLINE/GsGLINE |
| *ot* | Pointer to OT |
| *pri* | Position in OT |

### Explanation

This function assigns the straight line indicated by *lp* to the ordering table indicated by *ot*.

The GsSortLine() function registers single-color straight lines in OT, and the GsSortGLine() function graded straight lines in OT.

### Return value

None.

### Remarks

**See also:**

## GsSortObject3

Assigns an object to the ordering table (for use with GsDOBJ3).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortObject3** (**objp*, **otp*, *shift*)
**GsDOBJ3** **objp*;
**GsOT** **otp*;
**long** *shift*;

### Arguments

| | |
|---|---|
| *objp* | Pointer to an object |
| *otp* | Pointer to OT |
| *shift* | Specifies how many bits the Z value must be shifted to the right when assigning an object to the OT. |

### Explanation

Performs perspective transformation and light source calculation for a three-dimensional object handled by GsDOBJ3, and creates a drawing command within the PMD format packet memory. Performs Z-sort of the drawing commands generated immediately afterwards and assigns them to the OT indicated by *otp*.

The accuracy of Z may be adjusted with the value of *shift*. The maximum size of the ordering table (resolution) is 14 bits, but if this value is set to 12 bits, for example, the shift value must be set at 2 (=14-12), so that it will not be larger than the ordering table area.

### Return value

None

### Remarks

**See also:**

## GsSortObject4

Assigns an object to the ordering table (for use with GsDOBJ2).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortObject4** (**objp*, **otp*, *shift*, **scratch*)
**GsDOBJ2** **objp*;
**GsOT** **otp*;
**long** *shift*;
**unsigned long** **scratch*;

### Arguments

| | |
|---|---|
| *objp* | Pointer to an object |
| *otp* | Pointer to OT |
| *shift* | Specifies how many bits the Z value must be shifted to the right when assigning an object to the OT. |
| *scratch* | Pointer to the address of the scratch pad. |

### Explanation

Performs perspective transformation and light source calculation for a three-dimensional object to be handled by GsDOBJ2, and creates a drawing command within the packet area specified by GsSetWorkBase(). Performs Z-sort of the drawing commands generated immediately afterwards and assigns them to the OT indicated by *otp*.

The accuracy of Z may be adjusted with the value of *shift*. The maximum size of the ordering table (resolution) is 14 bits. If this value is set to 12 bits, for example, the shift value must be set at 2 (=14-12), so that it will not be larger than the ordering table area.

*scratch* is the specified scratchpad address used as work when automatic division is being performed. The scratchpad runs for 256 words from 0x1f800000 in cache memory.

To use the GsOBJ2 member attribute to enable division, perform an OR operation on the macros GsDIV1 through GsDIV5 (defined in libgs.h). For GsDIV1, a single polygon will be divided into four segments of 2 x 2. For GsDIV5, a single polygon will be divided into 1,024 segments of 32 x 32.

### Return value

None.

### Remarks

**See also:**

## GsSortObject4J

Allocation of object to ordering table (Function TABLE version.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.2* | *7/31/96* |

### Syntax

**void GsSortObject4J** (**objp*, **otp*, *shift*, **scratch*)
**GsDOBJ2** **objp*;
**GsOT** **otp*;
**long** *shift*;
**unsigned long** **scratch*;

### Arguments

| | |
|---|---|
| *objp* | Pointer to an object |
| *otp* | Pointer to OT |
| *shift* | Specifies how many bits the Z value must be shifted to the right when assigning an object to the OT. |
| *scratch* | Pointer to the address of the scratch pad |

### Explanation

When all the insignificant functions have been registered, this function's features are equal to those of GsSortObject4 (). In addition, for the programmer to be able to control the functions registered to the table, he can increase the code efficiency by taking care not to call the unnecessary insignificant functions.

GsSortObject4 () is used for prototyping, but ultimately memory can be saved if you switch to GsSortObject4J ().

A maximum of 40kbytes can be saved.

If 'dmy' is written first of all at the head of the function name for slots which do not register GsFCALL4, even if by chance that insignificant function is called, it will not cause a BUS ERROR and since the function name used when the function was first called is printed out, delete the 'dmy' and register.

GsFCALL4 GsSortObject4J ( ) reference function table.

### Return value

None.

### Remarks

**See also:**

## GsSortObject5

Assigns an object to the ordering table (for use with GsDOBJ5).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.0* | *7/31/96* |

### Syntax

**void GsSortObject5** (**objp*, **otp*, *shift*, **scratch*)
**GsDOBJ2** **objp*;
**GsOT** **otp*;
**long** *shift*;
**unsigned long** **scratch*;

### Arguments

| | |
|---|---|
| *objp* | Pointer to an object |
| *otp* | Pointer to OT |
| *shift* | Specifies how many bits the Z value must be shifted to the right when assigning an object to the OT. |
| *scratch* | Pointer to the address of the scratch pad. |

### Explanation

Performs transparency transformation and light source calculation for a three-dimensional object to be handled by GsDOBJ5, and creates in the preset packet area drawing commands that do not divide, and in the packet area specified by GsSetWorkBase() those drawing commands that do divide. Performs Z-sort of the drawing commands generated immediately afterwards and assigns them to the OT indicated by *otp*.

The accuracy of Z may be adjusted using the *shift* value. The maximum size of the ordering table (resolution) is 14 bits. If this value is set to 12 bits, for example, the shift value must be set at 2 (=14-12), so that it will not be larger than the ordering table area.

*scratch* is used as work when automatic division is being performed. To use attribute to enable division, perform an OR operation on the macros GsDIV1-GsDIV5 of libgs.h. For GsDIV1, a single polygon will be divided into four segments of 2 x 2. For GsDIV5, a single polygon will be divided into 1, 024 segments of 32 x 32.

*scratch* is the specified scratchpad address used as work when automatic division is being performed. The scratchpad runs for 256 words from 0x1f800000 in cache memory.

To use the GsOBJ2 member attribute to enable division, perform an OR operation on the macros GsDIV1 through GsDIV5 (defined in libgs.h). For GsDIV1, a single polygon will be divided into four segments of 2 x 2. For GsDIV5, a single polygon will be divided into 1,024 segments of 32 x 32.

### Return value

None

### Remarks

**See also:**

## GsSortObject5J

Assigns an object to the ordering table.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.2* | *7/31/96* |

### Syntax

**void GsSortObject5J** (**objp*, **otp*, *shift*, **scratch*)
**GsDOBJ2** **objp*;
**GsOT** **otp*;
**long** *shift*;
**unsigned long** **scratch*;

### Arguments

| | |
|---|---|
| *objp* | Pointer to an object |
| *otp* | Pointer to OT |
| *shift* | Specifies how many bits the Z value must be shifted to the right when assigning an object to the OT. |
| *scratch* | Pointer to the address of the scratch pad |

### Explanation

When all the insignificant functions have been registered, this function's features are equal to those of GsSortObject5 ( ) . In addition, for the programmer to be able to control the functions registered to the table, he can increase the code efficiency by taking care not to call the unnecessary insignificant functions.

GsSortObject5( ) is used for prototyping, but ultimately memory can be saved if you switch to GsSortObject5J( ).

A maximum of 40kbytes can be saved.

If 'dmy' is written first of all at the head of the function name for slots which do not register GsFCALL5, even if by chance that insignificant function is called, it will not cause a BUS ERROR and since the function name used when the function was first called is printed out, delete the 'dmy' and register.

GsFCALL5    GsSortObject5J ( ) reference function table

### Return value

None.

### Remarks

**See also:**

## GsSortOt

Assigns an OT to another OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *3.3* | *7/31/96* |

### Syntax

**GsOT** ***GsSortOt** (**ot_src*, **ot_dest*)
**GsOT** **ot_src*;
**GsOT** **ot_dest*;

### Arguments

| | |
|---|---|
| *ot_src* | Pointer to source OT |
| *ot_dest* | Pointer to destination OT |

### Explanation

This function assigns the OT given by *ot_src* to *ot_dest*. The representative value in the point field for each OT is used as the OTZ value. The integrated OT is inserted into *ot_dest*.

### Return value

Pointer to the integrated OT.

### Remarks

**See also:**

## GsSortPoly

Registers polygon drawing primitive in the OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSortPoly** (**prim*, **ot*, *pri*)
**void** **prim*;
**GsOt** **ot*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *prim* | Pointer to drawing primitive |
| *ot* | Pointer to OT |
| *pri* | Location in OT |

### Explanation

This function assigns the drawing primitive given by *prim* to the ordering table given by *ot*.

Out of the primitives defined by libgpu, the drawing primitive refers only to (POLY_....).

libgs requires no double buffering, since the contents of the primitive structure are copied in the packet generation area. Drawing coordinate values match the drawing coordinate system handled by libgs.

### Return value

None.

### Remarks

**See also:**

## GsSortSprite, GsSortFastSprite, GsSortFlipSprite

Registers a Sprite in the OT.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSortSprite** (**sp*, **otp*, *pri*)
**GsSPRITE** **sp*;
**GsOT** **otp*;
**unsigned short** *pri*;

**void GsSortFastSprite** (**sp*, **otp*, *pri*)
**GsSPRITE** **sp*;
**GsOT** **otp*;
**unsigned short** *pri*;

**void GsSortFlipSprite** (**sp*, **otp*, *pri*)
**GsSPRITE** **sp*;
**GsOT** **otp*;
**unsigned short** *pri*;

### Arguments

| | |
|---|---|
| *sp* | Pointer to a Sprite |
| *otp* | Pointer to OT |
| *pri* | Position in OT |

### Explanation

This function assigns the Sprite given by *sp* to the ordering table provided by *otp*.

All the parameters including Sprite indication locations are given by the sp members.

*pri* refers to the priority of the Sprite in the ordering table. The highest priority value is zero, with the lowest value depending on the size of the ordering table. Values beyond the size of the ordering table are clipped to the maximum ordering table value.

The GsSortFastSprite function provides high-speed processing, though enlargement, rotation, and reduction cannot be used. The Sprite structure members *nx*, *my*, *scalex*, *scaley*, and *rotate* are ignored.

GsSortFlipSprite() does not use the enlargement / rotation / reduction functions, and only supports flipping.

### Return value

None.

### Remarks

**See also:**

## GsSwapDispBuffer

Swaps double buffers.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgs.lib* | *Libgs.h* | *2.x* | *7/31/96* |

### Syntax

**void GsSwapDispBuffer** (*void*)

### Arguments

None.

### Explanation

This function exchanges the display buffer with the drawing buffer according to double buffer data set by GsSetDefDispBuffer(). Normally, swapping is done immediately after beginning vertical blanking.

This function performs the following:

- Sets display starting address
- Cancels blanking
- Sets double buffer index
- Switches two-dimensional clipping
- Sets libgte or libgpu offset
- Sets libgs offset

Note: The double buffer is implemented by offset. Using the GsOFSGPU or GsOFSGTE macro for the third argument of GsInitGraph() determines whether the libgte or libgpu offset should be set.

### Return value

None.

### Remarks

This function does not execute correctly when GPU drawing is in progress, so it is necessary to call this function after terminating drawing using ResetGraph (1).

**See also:**

## GsTMDdivF3L, GsTMDdivF3LFG, GsTMDdivF3NL, GsTMDdivNF3

TMD data flat triangle processing.

GsTMDdivF3L: flat triangle (light source calculation)

GsTMDdivF3LFG: flat triangle (light source calculation + FOG)

GsTMDdivF3NL: flat triangle (without light source calculation)

GsTMDdivNF3: flat triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivF3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivF3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivF3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivNF3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_F3** **primtop*;
(**TMD_P_NF3** **primtop*;)
**SVECTOR** *vertop;
**SVECTOR** **nortop*;
**POLY_F3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivNF3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) flat triangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivF3L(), Gs TMDdivF3LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–8**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |
| 5 | 32 x 32 division |

## Return value

Updated GPU packet buffer address.

## Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDdivF4L, GsTMDdivF4LFG, GsTMDdivF4NL, GsTMDdivNF4

TMD data flat quadrilateral processing.

GsTMDdivF4L: flat rectangle (light source calculation)

GsTMDdivF4LFG: flat rectangle (light source calculation + FOG)

GsTMDdivF4NL: flat rectangle (without light source calculation)

GsTMDdivNF4: flat rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivF4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivF4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivF4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivNF4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_F4** **primtop*;
(**TMD_P_NF4** **primtop*;)
**SVECTOR** *vertop;
**SVECTOR** **nortop*;
**POLY_F4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivNF4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) flat rectangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivF4L(), Gs TMDdivF4LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–9**

| ndiv value | processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |

| | |
|---|---|
| 5 | 32 x 32 division |

**Return value**

Updated GPU packet buffer address.

**Remarks**

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDdivG3L, GsTMDdivG3LFG, GsTMDdivG3NL, GsTMDdivNG3

TMD data Gouraud-shaded, triangle processing.

GsTMDdivG3L: Gouraud triangle (light source calculation)

GsTMDdivG3LFG: Gouraud triangle (light source calculation + FOG)

GsTMDdivG3NL: Gouraud triangle (without light source calculation)

GsTMDdivNG3: Gouraud triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivG3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivG3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivG3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivNG3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_G3** **primtop*;
(**TMD_P_NG3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_G3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** *otp;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivNG3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) Gouraud triangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivG3L(), Gs TMDdivG3LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–10**

| ndiv value | processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |

| | |
|---|---|
| 5 | 32 x 32 division |

**Return value**

Updated GPU packet buffer address.

**Remarks**

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDdivG4L, GsTMDdivG4LFG, GsTMDdivG4N, GsTMDdivNG4

TMD data Gouraud-shaded, quadrilateral processing.

GsTMDdivG4L: Gouraud rectangle (light source calculation)

GsTMDdivG4LFG: Gouraud rectangle (light source calculation + FOG)

GsTMDdivG4NL: Gouraud rectangle (without light source calculation)

GsTMDdivNG4: Gouraud rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivG4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivG4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivG4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivNG4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_G4** **primtop*;
(**TMD_P_NG4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_G4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** *otp;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivNG4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) Gouraud rectangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivG4L(), Gs TMDdivG4LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–11**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |

| | |
|---|---|
| 5 | 32 x 32 division |

**Return value**

Updated GPU packet buffer address.

**Remarks**

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDdivTF3L, GsTMDdivTF3LFG, GsTMDdivTF3NL, GsTMDdivTNF3

TMD data flat, textured triangle processing.

GsTMDdivTF3L: flat textured triangle (light source calculation)

GsTMDdivTF3LFG: flat textured triangle (light source calculation + FOG)

GsTMDdivTF3NL: flat textured triangle (without light source calculation)

GsTMDdivTNF3: flat textured triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivTF3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTF3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTF3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTNF3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_TF3** **primtop*;
(**TMD_P_TNF3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_FT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivTNF3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) flat textured triangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivTF3L(), Gs TMDdivTF3LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–12**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |

| | |
|---|---|
| 5 | 32 x 32 division |

## Return value

Updated GPU packet buffer address.

## Remarks

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDdivTF4L, GsTMDdivTF4LFG, GsTMDdivTF4NL, GsTMDdivTNF4

TMD data flat, textured quadrilateral processing.

GsTMDdivTF4L: flat textured rectangle (light source calculation)

GsTMDdivTF4LFG: flat textured rectangle (light source calculation + FOG)

GsTMDdivTF4NL: flat textured rectangle (without light source calculation)

GsTMDdivTNF4: flat textured rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivTF4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTF4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTF4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTNF4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_TF4** **primtop*;
(**TMD_P_TNF4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_FT4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivTNF4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) flat textured rectangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivTF4L(), Gs TMDdivTF4LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–13**

| ndiv value | processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |

| | |
|---|---|
| 5 | 32 x 32 division |

**Return value**

Updated GPU packet buffer address.

CAUTION

**Remarks**

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDdivTG3L, GsTMDdivTG3LFG, GsTMDdivTG3NL, GsTMDdivTNG3

TMD data Gouraud-shaded, textured triangle processing.

GsTMDdivTG3L: Gouraud texture triangle (light source calculation)

GsTMDdivTG3LFG: Gouraud texture triangle (light source calculation + FOG)

GsTMDdivTG3NL: Gouraud texture triangle (without light source calculation)

GsTMDdivTNG3: Gouraud texture triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivTG3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTG3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTG3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTNG3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_TG3** **primtop*;
(**TMD_P_TNG3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;
**DIVPOLYGON3** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivTNG3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) Gouraud texture triangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivTG3L(), Gs TMDdivTG3LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–14**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |

| | |
|---|---|
| 5 | 32 x 32 division |

**Return value**

Updated GPU packet buffer address.

**Remarks**

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDdivTG4L, GsTMDdivTG4LFG, GsTMDdivTG4NL, GsTMDdivTNG4

TMD data Gouraud-shaded, textured quadrilateral processing.

GsTMDdivTG4L: Gouraud texture rectangle (light source calculation)

GsTMDdivTG4LFG: Gouraud texture rectangle (light source calculation + FOG)

GsTMDdivTG4NL: Gouraud texture rectangle (without light source calculation)

GsTMDdivTNG4: Gouraud texture rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDdivTG4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTG4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTG4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*, **divp*)
**u_long** ***GsTMDdivTNG4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*, **divp*)
**TMD_P_TG4** **primtop*;
(**TMD_P_TNG4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;
**DIVPOLYGON4** **divp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDdivTNG4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |
| *divp* | Pointer to division work area |

### Explanation

For n (number of) Gouraud texture rectangles linked to the TMD file, this function divides polygons based on the *divp* -> ndiv value, performs coordinate transformation, perspective transformation, normal line clipping, display screen clipping, and light source calculation (GsTMDdivTG4L(), Gs TMDdivTG4LFG()), for the divided polygons, and then completes the GPU packet in the buffer and links to OT.

The *divp* -> ndiv values and division format are shown below:

**Table 8–15**

| ndiv value | Processing |
|---|---|
| 1 | 2x2 division |
| 2 | 4x4 division |
| 3 | 8x8 division |
| 4 | 16 x 16 division |

| | |
|---|---|
| 5 | 32 x 32 division |

**Return value**

Updated GPU packet buffer address.

**Remarks**

You must set *divp* -> ndiv (number of divisions) and *divp* -> pih.piv (display screen (clipping) resolution) beforehand.

**See also:**

## GsTMDfastF3GL

Flat graduation triangle (light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET *GsTMDfastF3GL** (**TMD_P_F3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

These functions perform coordinate transformation, perspective transformation, backface clipping, and light source calculation, for n (number of) triangles, create the GPU packet in the buffer, and links to OT.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsTMDfastF3GLFG

Flat graduation triangle (light source calculation + FOG).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET *GsTMDfastF3GLFG** (**TMD_P_F3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, *i***nt** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

These functions perform coordinate transformation, perspective transformation, backface clipping, and light source calculation, for n (number of) triangles, create the GPU packet in the buffer, and links to OT.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

### Remarks

**See also:**

## GsTMDfastF3GNL

Flat graduation triangle (without light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET *GsTMDfastF3GL**(**TMD_P_F3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, *i***nt** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

These functions perform coordinate transformation, perspective transformation, backface clipping, and light source calculation, for n (number of) triangles, create the GPU packet in the buffer, and links to OT.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsTMDfastF3L, GsTMDfastF3LFG, GsTMDfastF3NL, GsTMDfastNF3

TMD data flat triangle processing.

GsTMDfastF3L: flat triangle (light source calculation)

GsTMDfastF3LFG: flat triangle (light source calculation + FOG)

GsTMDfastF3NL: flat triangle (without light source calculation)

GsTMDfastNF3: flat triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastF3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastF3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastF3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastNF3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_F3** **primtop*;
(**TMD_P_NF3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_F3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastNF3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsTMDfastF3L(), GsTMDfastF3LFG()), for n (number of) flat triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastF3M

Flat triangle (light source calculation material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET** * **GsTMDfastF3M** (**TMD_P_F3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastF3MFG

Flat triangle (light source calculation FOG+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastF3MFG** (**TMD_P_F3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastF4L, GsTMDfastF4LFG, GsTMDfastF4NL, GsTMDfastNF4

TMD data flat quadrilateral processing.

GsTMDfastF4L: flat rectangle (light source calculation)

GsTMDfastF4LFG: flat rectangle (light source calculation + FOG)

GsTMDfastF4NL: flat rectangle (without light source calculation)

GsTMDfastNF4: flat rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastF4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastF4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastF4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastNF4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_F4** **primtop*;
(**TMD_P_NF4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_F4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastNF4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation(GsTMDfastF4L(), GsTMDfastF4LFG()), for n (number of) flat rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastF4M

TMD data flat texture quadrilateral processing (light source calculation+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET** * **GsTMDfastTF4MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

To use this function it must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Updated GPU packet buffer address.

**Table 8–16: External variables summary**

| Name | Type | Explanation |
|---|---|---|
| CLIP2 | RECT | Two dimension clipping area. Also set by GsClip2D |
| PSDMBASEX[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDMBASEY[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDIDX | u_short | Double buffer index |
| PSDCNT | u_long | Numbers are incremented by frame switch |
| POSITION | _GsPOSITION | 2 dimension offset |
| GsDRAWENV | DRAWENV | GS drawing environment |
| GsDISPENV | DISPENV | Gs display environment |
| GsLSMATRIX | MATRIX | Gs local screen matrix. Set by GsSetLs () |
| GsWSMATRIX | MATRIX | Gs World screen matrix. Set by GsSetRefView () and others |
| GsLIGHT_MODE | int | Default line mode |
| HWD0 | u_long | Horizontal resolution |
| VWD0 | u_long | Vertical resolution |
| GsLIGHTWSMATRIX | MATRIX | Gs write matrix.  Set by GsSetFlatLight () |
| GsIDMATRIX | MATRIX | Unit queue |
| GsIDMATRIX2 | MATRIX | Unit queue (Includes aspect conversion) |
| GsLIGHT_FUNC | Function pointer | Pointer for default light source calculation routine function used by GsDOBJ1, GsDOBJ2 |
| GsOUT_PACKET_P | u_long | Pointer to hold top of packet area. Set by GsSetWorkBase () |

| | | |
|---|---|---|
| GsMATE_C | u_long | Result when attribute is decoded (attenuation coefficient) |
| GsLMODE | u_long | Result when attribute is decoded (write mode) |
| GsLIGNR | u_long | Result when attribute is decoded (ignore write) |
| GsNDIV | u_long | Result when attribute is decoded (fragmentation number) |
| GsTRATE | u_long | Result when attribute is decoded (semi-transparent rate) |
| GsTON | u_long | Result when attribute is decoded (semi-transparent) |
| GsDISPON | u_long | Result when attribute is decoded (display/non-display) |
| GsFCALL5 | Structure | GsSortObject5J () function table |
| GsFCALL4 | Structure | GsSortObject4J () function table |

## Remarks

**See also:**

## GsTMDfastF4MFG

TMD data flat texture quadrilateral processing (light source calculation+material lighting+FOG).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET** * **GsTMDfastTF4MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

To use this function it must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Updated GPU packet buffer address.

**Table 8–17: External variables summary**

| Name | Type | Explanation |
|---|---|---|
| CLIP2 | RECT | Two dimension clipping area. Also set by GsClip2D |
| PSDMBASEX[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDMBASEY[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDIDX | u_short | Double buffer index |
| PSDCNT | u_long | Numbers are incremented by frame switch |
| POSITION | _GsPOSITION | 2 dimension offset |
| GsDRAWENV | DRAWENV | GS drawing environment |
| GsDISPENV | DISPENV | Gs display environment |
| GsLSMATRIX | MATRIX | Gs local screen matrix. Set by GsSetLs () |
| GsWSMATRIX | MATRIX | Gs World screen matrix. Set by GsSetRefView () and others |
| GsLIGHT_MODE | int | Default line mode |
| HWD0 | u_long | Horizontal resolution |
| VWD0 | u_long | Vertical resolution |
| GsLIGHTWSMATRIX | MATRIX | Gs write matrix. Set by GsSetFlatLight () |
| GsIDMATRIX | MATRIX | Unit queue |
| GsIDMATRIX2 | MATRIX | Unit queue (Includes aspect conversion) |
| GsLIGHT_FUNC | Function pointer | Pointer for default light source calculation routine function used by GsDOBJ1, GsDOBJ2 |
| GsOUT_PACKET_P | u_long | Pointer to hold top of packet area. Set by GsSetWorkBase () |

| | | |
|---|---|---|
| GsMATE_C | u_long | Result when attribute is decoded (attenuation coefficient) |
| GsLMODE | u_long | Result when attribute is decoded (write mode) |
| GsLIGNR | u_long | Result when attribute is decoded (ignore write) |
| GsNDIV | u_long | Result when attribute is decoded (fragmentation number) |
| GsTRATE | u_long | Result when attribute is decoded (semi-transparent rate) |
| GsTON | u_long | Result when attribute is decoded (semi-transparent) |
| GsDISPON | u_long | Result when attribute is decoded (display/non-display) |
| GsFCALL5 | Structure | GsSortObject5J () function table |
| GsFCALL4 | Structure | GsSortObject4J () function table |

**Remarks**

**See also:**

## GsTMDfastG3GL

Gouraud graduation triangle (light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *GsTMDfastG3GL** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

These functions perform coordinate transformation, perspective transformation, backface clipping, and light source calculation, for n (number of) triangles, create the GPU packet in the buffer, and links to OT.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

### Remarks

**See also:**

## GsTMDfastG3GLFG

Gouraud graduation triangle (light source calculation + FOG).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *GsTMDfastG3GLFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

These functions perform coordinate transformation, perspective transformation, backface clipping, and light source calculation, for n (number of) triangles, create the GPU packet in the buffer, and links to OT.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

### Remarks

**See also:**

## GsTMDfastG3GNL

Gouraud graduation triangle (without light source calculation).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.2* | *7/31/96* |

### Syntax

**PACKET *GsTMDfastG3GNL** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

These functions perform coordinate transformation, perspective transformation, backface clipping, and light source calculation, for n (number of) triangles, create the GPU packet in the buffer, and links to OT.

### Return value

Top address of unused packet area.

### Remarks

A graduation triangle indicates a TMD polygon of which each vertex has different RGB.

**See also:**

## GsTMDfastG3L, GsTMDfastG3LFG, GsTMDfastG3NL, GsTMDfastNG3

TMD data Gouraud-shaded, triangle processing.

GsTMDfastG3L: Gouraud triangle (light source calculation)

GsTMDfastG3LFG: Gouraud triangle (light source calculation + FOG)

GsTMDfastG3NL: Gouraud triangle (without light source calculation)

GsTMDfastNG3: Gouraud triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastG3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastG3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastG3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastNG3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_G3** **primtop*;
(**TMD_P_NG3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_G3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastNG3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation(GsTMDfastG3L(), GsTMDfastG3LFG()), for n (number of) Gouraud triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastG3M

Gouraud triangle (light source calculation material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastG3M** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastG3MFG

Gouraud triangle (light source calculation FOG+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastG3MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastG4L, GsTMDfastG4LFG, GsTMDfastG4NL, GsTMDfastNG4

TMD data Gouraud-shaded, quadrilateral processing.

GsTMDfastG4L: Gouraud rectangle (light source calculation)

GsTMDfastG4LFG: Gouraud rectangle (light source calculation + FOG)

GsTMDfastG4NL: Gouraud rectangle (without light source calculation)

GsTMDfastNG4: Gouraud rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastG4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastG4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastG4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastNG4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_G4** **primtop*;
(**TMD_P_NG4** **primtop*
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_G4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastNG4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsTMDfastG4L(), GsTMDfastG4LFG()), for n (number of) Gouraud rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastG4M

Gouraud square (light source calculation material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastG4M** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastG4MFG

Gouraud square (light source source calculation FOG+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastG4MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastTF3L, GsTMDfastTF3LFG, GsTMDfastTF3NL, GsTMDfastTNF3

TMD data flat, textured triangle processing.

GsTMDfastTF3L: flat textured triangle (light source calculation)

GsTMDfastTF3LFG: flat textured triangle (light source calculation + FOG)

GsTMDfastTF3NL: flat textured triangle (without light source calculation)

GsTMDfastTNF3: flat textured triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastTF3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTF3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTF3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTNF3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TF3** **primtop*;
(**TMD_P_TNF3** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_FT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastTNF3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsTMDfastTF3L(), GsTMDfastTF3LFG()), for n (number of) flat textured triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastTF3M

Flat texture triangle (light source calculation material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastTF3M** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastTF3MFG

Flat texture triangle (light source calculation FOG+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET** * **GsTMDfastTF3MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastTF4L, GsTMDfastTF4LFG, GsTMDfastTF4NL, GsTMDfastTNF4

TMD data flat, textured quadrilateral processing.
GsTMDfastTF4L: flat textured rectangle (light source calculation)
GsTMDfastTF4LFG: flat textured rectangle (light source calculation + FOG)
GsTMDfastTF4NL: flat textured rectangle (without light source calculation)
GsTMDfastTNF4: flat textured rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastTF4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTF4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTF4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTNF4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TF4** **primtop*;
(**TMD_P_TNF4** **primtop*;
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_FT4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastTNF4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsTMDfastTF4L(), GsTMDfastTF4LFG()), for n (number of) flat textured rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastTF4LM*, GsTMDfastTF4LFGM*, GsTMDfastTF4NLM*, GsTMDfastTNF4M*, GsTMDfastTG4LM*, GsTMDfastTG4LFGM*, GsTMDfastTG4NLM*, GsTMDfastTNG4M*, GsTMDdivTF4LM*, GsTMDdivTF4LFGM*, GsTMDdivTF4NLM*, GsTMDdivTNF4M*, GsTMDdivTG4LM*, GsTMDdivTG4LFGM*, GsTMDdivTG4NLM*, GsTMDdivTNG4M*, GsA4divTF4LM*, GsA4divTF4LFGM*, GsA4divTF4NLM*, GsA4divTNF4M*, GsA4divTG4LM*, GsA4divTG4LFGM*, GsA4divTG4NLM*, GsA4divTNG4M*

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.6* | *12/10/96* |

| | |
|---|---|
| GsTMDfastTF4LM | mip-map Flat Texture Square (Light Source Calc.) |
| GsTMDfastTF4LFGM | mip-map Flat Texture Square (Light Source Calc.Å{FOG) |
| GsTMDfastTF4NLM | mip-map Flat Texture Square (Without Light Source Calc.) |
| GsTMDfastTNF4M | mip-map Flat Texture Square (Without Light Source Calc.) |
| GsTMDfastTG4LM | mip-map Gouraud Texture Square(Light Source Calc.) |
| GsTMDfastTG4LFGM | mip-map Gouraud Texture Square(Light Source Calc.Å{FOG) |
| GsTMDfastTG4NLM | mip-map Gouraud Texture Square(Without Light Source Calc.) |
| GsTMDfastTNG4M | mip-map Gouraud Texture Square(Without Light Source Calc.) |
| GsTMDdivTF4LM | mip-map Flat Texture Square(Fixed DivisionÅ{Light Source Calc.) |
| GsTMDdivTF4LFGM | mip-map Flat Texture Square (Fixed DivisionÅ{Light Source Calc.Å{FOG) |
| GsTMDdivTF4NLM | mip-map Flat Texture Square (Fixed DivisionÅ{Without Light Source Calc.) |
| GsTMDdivTNF4M | mip-map Flat Texture Square (Fixed DivisionÅ{Without Light Source Calc.) |
| GsTMDdivTG4LM | mip-map Gouraud Texture Square (Fixed DivisionÅ{Light Source Calc.) |
| GsTMDdivTG4LFGM | mip-map Gouraud Texture Square (Fixed DivisionÅ{Light Source Calc.Å{FOG) |
| GsTMDdivTG4NLM | mip-map Gouraud Texture Square (Fixed DivisionÅ{Without Light Source Calc.) |
| GsTMDdivTNG4M | mip-map Gouraud Texture Square (Fixed DivisionÅ{Without Light Source Calc.) |
| GsA4divTF4LM | mip-map Flat Texture Square (Automatic DivisionÅ{Light Source Calc.) |
| GsA4divTF4LFGM | mip-map Flat Texture Square (Automatic DivisionÅ{Light Source Calc.Å{FOG) |
| GsA4divTF4NLM | mip-map Flat Texture Square (Automatic DivisionÅ{Without Light Source Calc.) |
| GsA4divTNF4M | mip-map Flat Texture Square (Automatic DivisionÅ{Without Light Source Calc.) |
| GsA4divTG4LM | mip-map Gouraud Texture Square (Automatic DivisionÅ{Light Source Calc.) |
| GsA4divTG4LFGM | mip-map Gouraud Texture Square (Automatic DivisionÅ{Light Source Calc.Å{FOG) |
| GsA4divTG4NLM | mip-map Gouraud Texture Square (Automatic DivisionÅ{Without Light Source Calc.) |
| GsA4divTNG4M | mip-map Gouraud Texture Square (Automatic DivisionÅ{Without Light Source Calc.) |

### Syntax

PACKET *GsTMDfastTF4LM(TMD_P_TF4 *op, VERT *vp, VERT *np,PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTF4LFGM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTF4NLM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTNF4M(TMD_P_TF4 *op, VERT *vp,  PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTG4LM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTG4LFGM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTG4NLM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTNG4M(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot)

PACKET *GsTMDdivTF4LM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp)
PACKET *GsTMDdivTF4LFGM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp)
PACKET *GsTMDdivTF4NLM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp) PACKET *GsTMDdivTNF4M(TMD_P_TF4 *op, VERT *vp, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp)
PACKET *GsTMDdivTG4LM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp)
PACKET *GsTMDdivTG4LFGM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp)
PACKET *GsTMDdivTG4NLM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp)
PACKET *GsTMDdivTNG4M(TMD_P_TG4 *op, VERT *vp, PACKET *pk,int n,int shift, GsOT *ot, DIVPOLYGON4 *divp)
PACKET *GsA4divTF4LM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTF4LFGM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTF4NLM(TMD_P_TF4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTNF4M(TMD_P_TF4 *op, VERT *vp, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTG4LM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTG4LFGM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTG4NLM(TMD_P_TG4 *op, VERT *vp, VERT *np, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTNG4M(TMD_P_TG4 *op, VERT *vp, PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)

## Argument

| | |
|---|---|
| *op* | TMD PRIMITIVE Starting Address |
| *vp* | TMD VERTEXS Starting Address |
| *np* | TMD NORMAL Starting Address |
| *pk* | GPU Packet Buffer Starting Address |
| *n* | Number of PRIMITIVEs |
| *shift* | Number of bits to be shifted when sorting to OT |
| *ot* | Pointer to GsOT |
| *scratch* | Non-used scratch pad Starting Address |

## Explanation

Low level function group of GsSortObject4J()

Need to be register into GsFCALL4 as a low level function before using.

This function performs mip-map, texture switching based on the polygon size , to the flat texture squares included in the TMD data.

Locate texture on the V-RAM as below;

```
+--------------+-------+
|              |       |
|              |       |
|              |       |
|              +---+---+
|              | +-+   |
|              | +-+---+
|              |
+--------------+
```

There are four texture sizes,  1, 1/4, 1/16, and 1/64.

Which texture size should be used is determined by the polygon outer product.

Polygon vertices must be in the order below;

```
0        1
+--------+
|        |
|        |
|        |
|        |
+--------+
2        3
```

Return Value  Non-used Packet Area Starting Address

**See also:**

## GsTMDfastTF4M

TMD data flat texture quadrilateral processing (light source calculation+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET** * **GsTMDfastTF4MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

To use this function it must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Updated GPU packet buffer address.

**Table 8–18: External variables summary**

| Name | Type | Explanation |
|---|---|---|
| CLIP2 | RECT | Two dimension clipping area. Also set by GsClip2D |
| PSDMBASEX[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDMBASEY[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDIDX | u_short | Double buffer index |
| PSDCNT | u_long | Numbers are incremented by frame switch |
| POSITION | _GsPOSITION | 2 dimension offset |
| GsDRAWENV | DRAWENV | GS drawing environment |
| GsDISPENV | DISPENV | Gs display environment |
| GsLSMATRIX | MATRIX | Gs local screen matrix. Set by GsSetLs () |
| GsWSMATRIX | MATRIX | Gs World screen matrix. Set by GsSetRefView () and others |
| GsLIGHT_MODE | int | Default line mode |
| HWD0 | u_long | Horizontal resolution |
| VWD0 | u_long | Vertical resolution |
| GsLIGHTWSMATRIX | MATRIX | Gs write matrix. Set by GsSetFlatLight () |
| GsIDMATRIX | MATRIX | Unit queue |
| GsIDMATRIX2 | MATRIX | Unit queue (Includes aspect conversion) |
| GsLIGHT_FUNC | Function pointer | Pointer for default light source calculation routine function used by GsDOBJ1, GsDOBJ2 |
| GsOUT_PACKET_P | u_long | Pointer to hold top of packet area. Set by GsSetWorkBase () |

| | | |
|---|---|---|
| GsMATE_C | u_long | Result when attribute is decoded (attenuation coefficient) |
| GsLMODE | u_long | Result when attribute is decoded (write mode) |
| GsLIGNR | u_long | Result when attribute is decoded (ignore write) |
| GsNDIV | u_long | Result when attribute is decoded (fragmentation number) |
| GsTRATE | u_long | Result when attribute is decoded (semi-transparent rate) |
| GsTON | u_long | Result when attribute is decoded (semi-transparent) |
| GsDISPON | u_long | Result when attribute is decoded (display/non-display) |
| GsFCALL5 | Structure | GsSortObject5J () function table |
| GsFCALL4 | Structure | GsSortObject4J () function table |

## Remarks

**See also:**

## GsTMDfastTF4MFG

TMD data flat texture quadrilateral processing (light source calculation+material lighting+FOG).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET** * **GsTMDfastTF4MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

To use this function it must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Updated GPU packet buffer address.

**Table 8–19: External variables summary**

| Name | Type | Explanation |
|---|---|---|
| CLIP2 | RECT | Two dimension clipping area. Also set by GsClip2D |
| PSDMBASEX[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDMBASEY[2] | u_short | Double buffer origin (X coordinate). Set by GsDefDispbuff () |
| PSDIDX | u_short | Double buffer index |
| PSDCNT | u_long | Numbers are incremented by frame switch |
| POSITION | _GsPOSITION | 2 dimension offset |
| GsDRAWENV | DRAWENV | GS drawing environment |
| GsDISPENV | DISPENV | Gs display environment |
| GsLSMATRIX | MATRIX | Gs local screen matrix. Set by GsSetLs () |
| GsWSMATRIX | MATRIX | Gs World screen matrix. Set by GsSetRefView () and others |
| GsLIGHT_MODE | int | Default line mode |
| HWD0 | u_long | Horizontal resolution |
| VWD0 | u_long | Vertical resolution |
| GsLIGHTWSMATRIX | MATRIX | Gs write matrix.  Set by GsSetFlatLight () |
| GsIDMATRIX | MATRIX | Unit queue |
| GsIDMATRIX2 | MATRIX | Unit queue (Includes aspect conversion) |
| GsLIGHT_FUNC | Function pointer | Pointer for default light source calculation routine function used by GsDOBJ1, GsDOBJ2 |
| GsOUT_PACKET_P | u_long | Pointer to hold top of packet area. Set by GsSetWorkBase () |

| | | |
|---|---|---|
| GsMATE_C | u_long | Result when attribute is decoded (attenuation coefficient) |
| GsLMODE | u_long | Result when attribute is decoded (write mode) |
| GsLIGNR | u_long | Result when attribute is decoded (ignore write) |
| GsNDIV | u_long | Result when attribute is decoded (fragmentation number) |
| GsTRATE | u_long | Result when attribute is decoded (semi-transparent rate) |
| GsTON | u_long | Result when attribute is decoded (semi-transparent) |
| GsDISPON | u_long | Result when attribute is decoded (display/non-display) |
| GsFCALL5 | Structure | GsSortObject5J () function table |
| GsFCALL4 | Structure | GsSortObject4J () function table |

## Remarks

**See also:**

## GsTMDfastTG3L, GsTMDfastTG3LFG, GsTMDfastTG3NL, GsTMDfastTNG3

TMD data Gouraud-shaded, textured triangle processing.GsTMDfastTG3L: Gouraud texture triangle (light source calculation)

GsTMDfastTG3LFG: Gouraud texture triangle (light source calculation + FOG)

GsTMDfastTG3NL: Gouraud texture triangle (without light source calculation)

GsTMDfastTNG3: Gouraud texture triangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastTG3L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTG3LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTG3NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTNG3** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TG3** **primtop*;
(**TMD_P_TNG3** **primtop*
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastTNG3(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsTMDfastTG3L() and GsTMDfast TG3LFG()), for n (number of) Gouraud texture triangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastTG3LFG_FLIP

Gouraud texture triangle (light source calculation FOG + normal line flip).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long *GsTMDfastTG3LFG_FLIP**(**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TG3** **primtop*;
**(TMD_P_TNG3** **primtop*;**)**
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to TMD primitive starting address |
| *(primtop)* | GsTMDfastTNG3 is a packet (normal line) |
| *vertop* | Pointer to TMD vertex starting address |
| *nortop* | Pointer to TMD normal starting address |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of process polygons |
| *shift* | Number of Z value bit to be right-shifted when allocating upon OT allocation |
| *otp* | Pointer to OT |

### Explanation

These functions perform coordinate conversion, perspective transformation, backface clip normal line flip,(light source calculation), generate GPU packet on the buffer, then link to OT (used in GsSortObject4).

### Return value

Updated GPU packet buffer address.

### Remarks

Please refer to libgs.

**See also:**

## GsTMDfastTG3L_FLIP

Gouraud texture triangle (light source calculation normal line flip).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long *GsTMDfastTG3L_FLIP**(**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp)*
**TMD_P_TG3** **primtop*;
**(TMD_P_TNG3** **primtop*;**)**
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to TMD primitive starting address |
| *(primtop)* | GsTMDfastTNG3 is a packet (normal line) |
| *vertop* | Pointer to TMD vertex starting address |
| *nortop* | Pointer to TMD normal starting address |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of process polygons |
| *shift* | Number of Z value bit to be right-shifted when allocating upon OT allocation |
| *otp* | Pointer to OT |

### Explanation

These functions perform coordinate conversion, perspective transformation, backface clip normal line flip,(light source calculation), generate GPU packet on the buffer, then link to OT (used in GsSortObject4).

### Return value

Updated GPU packet buffer address.

### Remarks

Please refer to libgs.

### Remarks

**See also:**

## GsTMDfastTG3M

Gouraud texture triangle (light source calculation material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastTG3M** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastTG3MFG

Gouraud texture triangle (light source calculation FOG+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastTG3MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastTG3NL_FLIP

Gouraud texture triangle (No light source calculation normal line flip).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long * GsTMDfastTG3NL_FLIP**(**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TG3** **primtop*;
**(TMD_P_TNG3** **primtop*;**)**
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to TMD primitive starting address |
| *(primtop)* | GsTMDfastTNG3 is a packet (normal line) |
| *vertop* | Pointer to TMD vertex starting address |
| *nortop* | Pointer to TMD normal starting address |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of process polygons |
| *shift* | Number of Z value bit to be right-shifted when allocating upon OT allocation |
| *otp* | Pointer to OT |

### Explanation

These functions perform coordinate conversion, perspective transformation, backface clip normal line flip,(light source calculation), generate GPU packet on the buffer, then link to OT (used in GsSortObject4).

### Return value

Updated GPU packet buffer address.

### Remarks

Please refer to libgs.

**See also:**

## GsTMDfastTG4L, GsTMDfastTG4LFG, GsTMDfastTG4NL, GsTMDfastTNG4

TMD data Gouraud-shaded, textured quadrilateral processing.

GsTMDfastTG4L: Gouraud texture rectangle (light source calculation)

GsTMDfastTG4LFG: Gouraud texture rectangle (light source calculation + FOG)

GsTMDfastTG4NL: Gouraud texture rectangle (without light source calculation)

GsTMDfastTNG4: Gouraud texture rectangle (without light source calculation)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long** ***GsTMDfastTG4L** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTG4LFG** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTG4NL** (**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp*)
**u_long** ***GsTMDfastTNG4** (**primtop*, **vertop*, **s*, *n*, *shift*, **otp*)
**TMD_P_TG4** **primtop*;
(**TMD_P_TNG4** **primtop*;)
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT4** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to top address of TMD PRIMITIVE<br>In GsTMDfastTNG4(), a packet without a normal line |
| *vertop* | Pointer to top address of TMD VERTEX |
| *nortop* | Pointer to top address of TMD NORMAL |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of target polygons |
| *shift* | Specifies which bit of Z value to shift to right when assigning OT. |
| *otp* | Pointer to OT |

### Explanation

This function performs coordinate transformation, perspective transformation, normal line clipping, and light source calculation (GsTMDfastTG4L() and GsTMDfast TG4 LFG()), for n (number of) Gouraud texture rectangles linked to the TMD file, completes the GPU packet in the buffer, and links to OT.

See libgs in the Library Overview manual for details.

### Return value

Updated GPU packet buffer address.

### Remarks

**See also:**

## GsTMDfastTG4M

Gouraud texture square (light source calculation material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET * GsTMDfastTG4M(TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastTG4MFG

Gouraud texture square (light source calculation FOG+material lighting).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.5* | *7/31/96* |

### Syntax

**PACKET** * **GsTMDfastTG4MFG** (**TMD_P_G3G** **op*, **VERT** **vp*, **VERT** **np*, **PACKET** **pk*, **int** *n*, **int** *shift*, **GsOT** **ot*, **u_long** **scratch*);

### Arguments

| | |
|---|---|
| *op* | Pointer to top address of TMD primitive |
| *vp* | Pointer to top address of TMD vertex |
| *np* | Pointer to top address of TMD normal |
| *pk* | Pointer to top address of GPU packet buffer |
| *n* | Number of primitives |
| *shift* | Number of bits to shift when assigning OT |
| *ot* | Pointer to GsOT |
| *scratch* | Pointer to top address of unused scratch-pad memory |

### Explanation

These functions are low-level functions of GsSortObject4J().

These must be registered to GsFCALL4 as low-level functions to be used.

Only when lighting mode is set to "material ON", the luminance is attenuated according to the parameter given as GsDOBJ2 attribute at light source calculation.

### Return value

Top address of unused packet area.

### Remarks

**See also:**

## GsTMDfastTNG3_FLIP

Gouraud texture triangle (No light source calculation normal line flip).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgte.lib* | *Libgte.h* | *3.0* | *7/31/96* |

### Syntax

**u_long * GsTMDfastTNG3_FLIP**(**primtop*, **vertop*, **nortop*, **s*, *n*, *shift*, **otp)*
**TMD_P_TG3** **primtop*;
**(TMD_P_TNG3** **primtop*;**)**
**SVECTOR** **vertop*;
**SVECTOR** **nortop*;
**POLY_GT3** **s*;
**u_long** *n*;
**u_long** *shift*;
**GsOT** **otp*;

### Arguments

| | |
|---|---|
| *primtop* | Pointer to TMD primitive starting address |
| *(primtop)* | GsTMDfastTNG3 is a packet (normal line) |
| *vertop* | Pointer to TMD vertex starting address |
| *nortop* | Pointer to TMD normal starting address |
| *s* | Pointer to GPU packet buffer address |
| *n* | Number of process polygons |
| *shift* | Number of Z value bit to be right-shifted when allocating upon OT allocation |
| *otp* | Pointer to OT |

### Explanation

These functions perform coordinate conversion, perspective transformation, backface clip normal line flip,(light source calculation), generate GPU packet on the buffer, then link to OT (used in GsSortObject4).

### Return value

Updated GPU packet buffer address.

### Remarks

Please refer to libgs.

**See also:**

# Chapter 9: CD/Streaming Library
## Table of Contents

## CdlATV

Audio attenuation structure.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Structure

**typedef struct {**
    **unsigned char** *val0*;
    **unsigned char** *val1*;
    **unsigned char** *val2*;
    **unsigned char** *val3*;
**} CdlATV**;

### Members

| | |
|---|---|
| *val0* | CD (L) --> SPU (L) reduction |
| *val1* | CD (L) --> SPU (R) reduction |
| *val2* | CD (R) --> SPU (L) reduction |
| *val3* | CD (R) --> SPU (R) reduction |

### Explanation

### Remarks

**See also:**

## CdlFILE

ISO-9660 file system file descriptor.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Structure

```
typedef struct {
    CdlLOC pos;
    unsigned long size;
    char name[16];
} CdlFILE;
```

### Members

| | |
|---|---|
| *pos* | File position |
| *size* | File size |
| *name* | File name |

### Explanation

Get position and size of ISO-9660 CD-ROM file.

### Remarks

**See also:**

## CdlFILTER

ADPCM channel.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
    **u_char** *file*;
    **u_char** *chan*;
    **u_short** *pad*;
**} CdlFILTER;**

### Members

| | |
|---|---|
| *file* | File ID |
| *chan* | Channel ID |
| *pad* | System reserved |

### Explanation

Sets the multi-channel ADPCM play channel.

### Remarks

**See also:** CdlSetfilter (p. ).

## cdlLoc

Time-code based CD-ROM disc position.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Structure

```
typedef struct {
    unsigned char minute;
    unsigned char second;
    unsigned char sector;
    unsigned char track;
} CdlLOC;
```

### Members

| | |
|---|---|
| *minute* | Minute |
| *second* | Second |
| *sector* | Sector |
| *track* | Track number |

### Explanation

Structure defining a time-code position on a CD-ROM. The time code is based on the time needed to reach that position when playing the disc from the beginning at normal speed.

### Remarks

The track member is not used at present.

**See also:**

## StHEADER

Sector header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Structure

**typedef struct {**
  **unsigned short** *id*;
  **unsigned short** *type*;
  **unsigned short** *secCount*;
  **unsigned short** *nSectors*;
  **unsigned long** *frameCount*;
  **unsigned long** *frameSize*;
  **unsigned short** *width*;
  **unsigned short** *height*;
  **unsigned long** *dummy1*;
  **unsigned long** *dummy2*;
  **CdlLoc** *loc*;
**} StHEADER;**

### Members

| | |
|---|---|
| *id* | Reserved by system |
| *type* | Data type (always 0x0160) |
| *secCount* | Sector offset within 1 frame |
| *nSectors* | Number of sectors comprising one frame |
| *frameCount* | Movie absolute frame number |
| *frameSize* | Movie data size (in long words) |
| *width* | Movie horizontal size |
| *height* | Movie vertical size |
| *dummy1* | Reserved by system |
| *dummy2* | Reserved by system |
| *loc* | File location |

### Explanation

Movie sector header.

If a header obtained with StGetNext() is written to this structure, the various items of information can be accessed through the structure members.

For details of information structure, refer to “Data Format” in the Run-time Library Overview manual.

### Remarks

**See also:**

## CdComstr

Get character string corresponding to command code (for debugging).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**char *CdComstr** (*com*)
**unsigned char** *com*;

### Arguments

| | |
|---|---|
| *code* | Command completion code |

### Explanation

For debugging. Get corresponding character string from processing status code. For example, get the following character strings for these codes.

**Table 9–1**

| Command Code | Character String |
|---|---|
| CdlNop | "CdlNop" |
| CdlSetloc | "CdlSetloc" |
| CdlPlay | "CdlPlay" |
| CdlForward | "CdlForward" |
| CdlBackword | "CdlBackword" |

### Return value

Pointer to start of character string.

### Remarks

**See also:**

## CdControl

Issue primitive command to CD-ROM system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdControl** (*com*, **param*, **result*)
**unsigned char** *com*, **param*, **result*;

### Arguments

| | |
|---|---|
| *com* | Command code |
| *param* | Pointer to command arguments |
| *result* | Pointer to return value storage buffer (requires 8 bytes) |

### Explanation

Issues the primitive command *com* to the CD-ROM system. If the command takes an argument, CdControl() sets these arguments in *param*. Uses result to store the return value of the command in the specified buffer.

The stored contents of command (*com*), the arguments (*param*), and the return value (*result*) are listed below.

This function is a non-blocking function, so it is necessary to use CdSync to detect actual transfer termination.

**Table 9–2: Primitive command overview**

| Symbol | Code | Type | Contents |
|---|---|---|---|
| CdlNop | 0x01 | B | NOP (No Operation) |
| CdlSetloc | 0x02 | B | Sets the seek target position |
| CdlPlay | 0x03 | B | Commence CD-DA play |
| CdlForward | 0x04 | B | Forward |
| CdlBackword | 0x05 | B | Rewind |
| CdlReadN | 0x06 | B | Start data read (with retry) |
| CdlStanby | 0x07 | N | Stand by with disk rotating |
| CdlStop | 0x08 | N | Disk stopped |
| CdlPause | 0x09 | N | Pause at current position |
| CdlMute | 0x0b | B | CD-DA mute |
| CdlDemute | 0x0c | B | Cancel mute |
| CdlSetfilter | 0x0d | B | Choose ADCPM play sector |
| CdlSetmode | 0x0e | B | Set basic mode |
| CdlGetlocL | 0x10 | B | Get logical location (data sector) |
| CdlGetlocP | 0x11 | B | Get physical location (audio sector) |
| CdlSeekL | 0x15 | N | Logical seek (data sector seek) |
| CdlSeekP | 0x16 | N | Physical seek (audio sector seek) |
| CdlReadS | 0x1b | B | Commence data read (no retry) |
| CdlReset | 0x1c | B | Reset |

B: Blocking, N: Non-Blocking operation

**Table 9–3: Primitive commands that take arguments and their arguments**

| Symbol | Parameter | Type | Contents |
|---|---|---|---|
| CdlSetloc | \|CdlLOC | * | Start sector location |
| CdlReadN | \|CdlLOC | * | Start sector location |
| CdlReadS | \|CdlLOC | * | Start sector location |
| CdlPlay | \|CdlLOC | * | Start sector location |
| CdlSetfilter | \|CdlFILTER | * | Set ADCPM sector play |
| CdlSetmode | \|u_char | * | Set basic mode |

**Table 9–4: Return values of primitive commands**

| Symbol | Return values and locations of the bytes where they are stored | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| CdlNop | Status | | | | | | | |
| CdlSetloc | Status | | | | | | | |
| CdlPlay | Status | | | | | | | |
| CdlForward | Status | | | | | | | |
| CdlBackword | Status | | | | | | | |
| CdlReadN | Status | | | | | | | |
| CdlStanby | Status | | | | | | | |
| CdlStop | Status | | | | | | | |
| CdlPause | Status | | | | | | | |
| CdlMute | Status | | | | | | | |
| CdlDemute | Status | | | | | | | |
| CdlSetfilter | Status | | | | | | | |
| CdlSetmode | Status | | | | | | | |
| CdlGetlocL | min | sec | sector | mode | file | chan | | |
| CdlGetlocP | track | index | min | sec | frame | amin | asec | aframe |
| CdlSeekL | Status | | | | | | | |
| CdlSeekP | Status | | | | | | | |
| CdlReadS | Status | | | | | | | |
| CdlReset | Status | | | | | | | |

## Return value

1 if the command is issued successfully. 0 if failed.

## Remarks

Set *param* to 0 for commands that do not require arguments. If result is set to 0, the return value is not stored.

## See also:

## CdControlB

Issue primitive command to CD-ROM system (Blocking-type function).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdControlB** (*com*, **param*, **result*)
**unsigned char** *com*, **param*, **result*;

### Arguments

| | |
|---|---|
| *com* | Command code |
| *param* | Pointer to command arguments |
| *result* | Pointer to return value storage buffer (requires 8 bytes) |

### Explanation

Issues the primitive command *com* to the CD-ROM system. If the command takes an argument, CdControlB() sets these arguments in *param*. Uses result to store the return value of the command in the specified buffer.

CdControlB() is identical to CdControl() except for the block function that waits to return until processing terminates.

For details, see the commands and arguments of CdControl(), and the Run-time Library 3.0 Overview manual.

### Return value

1 if issued successfully. 0 if failed.

### Remarks

Set *param* to 0 for commands that do not require arguments. If result is set to 0, the return value is not stored.

**See also:**

## CdControlF

Issue primitive command to CD-ROM system (highspeed type).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdControlF** (*com*, **param*)
**unsigned char** *com*, **param*;

### Arguments

| | |
|---|---|
| *com* | Command code (see separate item) |
| *param* | Pointer to an argument for command |

### Explanation

Issues the primitive command *com* to the CD-ROM system. If the command takes an argument, CdControlF() sets these arguments in *param*. Uses result to store the return value of the command in the specified buffer.

CdControlF() is fast because it does no handshaking with the subsystem (it does not even wait for command acknowledgement (ACK)).

For details, see the commands and arguments of CdControl(), and the Run-time Library 3.0 Overview.

### Return value

1 if issued successfully. 0 if failed.

### Remarks

Set *param* to 0 for commands that do not require arguments. At present 1 is always returned, so "return value" has no significance.

**See also:**

## CdDataCallback

Defines a routine that will be executed when a data transfer initiated by CdGetSector() is completed.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.0* | *7/31/96* |

### Syntax

**int CdDataCallback** (**func*)
**void** (**func*)()

### Arguments

| | |
|---|---|
| *func* | Pointer to address of callback function |

### Explanation

Defines a routine that will be executed when the data transfer of data initiated by the CdGetSector() function has been completed. The *func* parameter is the address of the desired routine. If *func* is 0, then any previous callback routine is disabled.

### Return Value

Address of previously set callback

### Remarks

While *func* is executing, subsequent data transfer complete interrupts are masked. Therefore, *func* should return as soon as the necessary processing is completed.

To restore the previous callback, preserve the return value and when processing finishes, use it to restore the previous callback address.

**See also:**

## CdDataSync

Waits for a data transfer initiated by CdGetSector() to be completed.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.2* | *7/31/96* |

### Syntax

**int CdDataSync** (*mode*)
**int** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Polling mode:<br>0: Blocking<br>1: Non-blocking |

### Explanation

Waits for the data transfer of data initiated by the CdGetSector() function to be completed. The *mode* parameter determines the method of polling. If *mode* is 0, then the function will wait for the data transfer to be completed. If *mode* is 1, then the function will poll the current status and return.

### Return Value

Returns 0 if transfer is completed. Returns 1 if transfer is still being performed. Returns -1 if an error occurred.

### Remarks

**See also:**

## CdDiskReady

Determine CD-ROM status after disc change.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**int CdDiskReady** (**int** *mode*)

### Arguments

| | |
|---|---|
| *mode* | Check mode<br>0: Blocking type<br>1: Non-blocking type |

### Explanation

This function checks the CD-ROM status after a disc change to determine whether a command can be issued. Immediately after a disc is changed, there is a delay of a few seconds during which commands may not be issued. This function checks the status so that your program knows when issuing a command is safe.

When the *mode* parameter is 0, this function waits until the CD-ROM status has stabilized and commands may be issued before returning. When the *mode* parameter is 1, this function simply returns the current status.

### Return value

| | |
|---|---|
| CdlComplete | The state where a command can be issued. |
| CdlDiskError | Blocking type:<br>No discs or defected disc.<br>Non-blocking type:<br>Not stable, no discs, or defected disc. |
| CdlStatShellOpen | Disc cover is open. |

### Remarks

It is recommended that your program use this function immediately after initiating a disc change to wait for the disc cover to be closed and the CD-ROM status to stabilize. After this is done, check the disc format using the CDGetDiskType() function and proceed accordingly.

**Note:**

Following is the maximum wait time required for returning from a blocking type function call:

DebuggingStation:

| | |
|---|---|
| CD-R | Maximum of 12 seconds |
| CD-DA | Maximum of 12 seconds |
| No disc | Approximately 5 seconds |

PlayStation:

| | |
|---|---|
| Black CD | Maximum of 3 seconds |
| CD-DA | Maximum of 5 seconds |
| No disc | Approximately 5 seconds |

Although non-blocking type function returns immediately after checking the disc status, it cannot differentiate two error cases, the non-stable status and no disc case. Thus it is recommended to manage the time out according to the wait time shown above.

**Note:**

This function does not operate correctly on the DTL-H2000 development system.

**See also:**

## CdFlush

CD-ROM.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.2* | *7/31/96* |

### Syntax

**void CdFlush ( void )**

### Arguments

None.

### Explanation

This function obtains the disc format. Currently only CD-ROM format can be recognized.

### Return value

| | |
|---|---|
| CdlCdromFormat | CD-ROM format |
| CdlOtherFormat | Other format |
| CdlStatShellOpen | Disc cover is open |
| CdlStatNoDisk | No discs |

### Remarks

On DebuggingStation, although PlayStation disc (black disc), CD-R, and other CD-ROM (ISO9600 format) can be recognized as a CD-ROM, on PlayStation (consumer model), only the PlayStation disc can be recognized as CD-ROM. CD-DA is always recognized as "Other Format".

**Note:**

Immediately after changing discs, it is recommended that your program call the CdDisKReady() function, followed by the CdDiskType() function.

**Note:**

This function does not operate correctly on the DTL-H2000 development system.

**See also:**

## CdGetDiskType

Obtains disc format.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**int CdGetDiskType** (**void**)

### Arguments

None.

### Explanation

This function obtains the disc format. Currently only CD-ROM format can be recognized.

### Return value

| | |
|---|---|
| CdlCdromFormat | CD-ROM format |
| CdlOtherFormat | Other format |
| CdlStatShellOpen | Disc cover is open |
| CdlStatNoDisk | No discs |

### Remarks

On DebuggingStation, although PlayStation disc (black disc), CD-R, and other CD-ROM (ISO9600 format) can be recognized as a CD-ROM, on PlayStation (consumer model), only the PlayStation disc can be recognized as CD-ROM. CD-DA is always recognized as "Other Format".

**Note:**

Immediately after changing discs, it is recommended that your program call the CdDisKReady() function, followed by the CdDiskType() function.

**Note:**

This function does not operate correctly on the DTL-H2000 development system.

**See also:**

## CdGetSector

Transfer up to one sector’s worth of data from the CD-ROM read buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdGetSector** (**buf, size*)
**void** **buf*;
**int** *size*;

### Arguments

| | |
|---|---|
| *buf* | Pointer to address of buffer that will receive data |
| *size* | Length of the data to transfer, maximum is 1 sector |

### Explanation

The CdGetSector() function is used to transfer sector buffer data to main memory. The *buf* parameter specifies the address where the data will be written. The *size* parameter specifies the length of the data that will be written.

### Return Value

1 if issued successfully, otherwise 0.

### Remarks

The CdGetSector() function is non-blocking. Use CdDataSync() or CdDataCallback() to determine if the transfer of data has been completed.

**See also:**

## CdGetToc

Read CD-ROM table of contents information.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdGetToc** (**loc*)
**CdlLOC** **loc*;

### Arguments

| | |
|---|---|
| *loc* | Pointer to location table |

### Explanation

Get starting position of each track on the CD-ROM disc.

### Return value

Positive integer returns a track number. Anything else returns Error.

### Remarks

The maximum number of tracks is 100.

**See also:**

## CdInit

Initializes CD-ROM system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void CdInit** (*mode*)
**CdInit** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Reset mode |

### Explanation

Resets the CD-ROM subsystem. The *mode* parameter is not used by current versions of the library and should be set to 0.

### Return value

None.

### Remarks

**See also:**

## CdInitFileSystem

Initializes CD-ROM file driver.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void CdInitFileSystem** (**int** *mode*)
**int** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Reset mode |

### Explanation

Resets the CD-ROM file system. The *mode* parameter is not used by current versions of the library and should be set to 0.

If files on the CD-ROM are accessed using the standard C library functions open(), read(), etc., then this function must be called first to initialize the file system.

### Return value

None

### Remarks

**See also:**

## CdIntstr

Get character string corresponding to command processing status (for debugging).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**char *CdIntstr** (*intr*)
**unsigned char** *intr*;

### Arguments

| | |
|---|---|
| *intr* | Processing status code |

### Explanation

For debugging. Get character string corresponding to processing status code *intr*. For debugging.

Table 9–5

| Processing Status | Character String |
|---|---|
| CdlNoIntr | "CdlNoIntr" |
| CdlComplete | "CdlComplete" |
| CdlDiskError | "CdlDiskError" |

### Return value

Pointer to start of character string.

### Remarks

**See also:**

## CdIntToPos

Translate CD position information from an absolute sector number to a minute/seconds/sector time code.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**CdlLOC *CdIntToPos** (*i*, **p*)
**int** *i*;
**CdlLOC** **p*;

### Arguments

| | |
|---|---|
| *i* | Absolute sector number |
| *p* | Pointer to a CdlLOC structure that will be set to the position time code |

### Explanation

Calculate value for minute/second/sector from absolute sector number.

### Return value

Pointer to *p*.

### Remarks

**See also:**

## CdLastCom

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.2* | *7/31/96* |

### Syntax

**int CdLastCom** (**void**)

### Arguments

None.

### Explanation

Returns the last command issued by CDControl/CDControlB.

Table 9–6

| Symbol | Code | Type | Details |
|---|---|---|---|
| CdlNop | 0x01 | B | NOP (No Operation) |
| CdlSetloc | 0x02 | B | Set seek packet location |
| CdlPlay | 0x03 | B | CD-DA start play |
| CdlForward | 0x04 | B | Fast Forward |
| CdlBackword | 0x05 | B | Rewind |
| CdlReadN | 0x06 | B | Data read start (with retry) |
| CdlStandby | 0x07 | N | Wait with disc rotating |
| CdlStop | 0x08 | N | Stop disc rotation |
| CdlPause | 0x09 | N | Temporarily stop at current location |
| CdlMute | 0x0b | B | CD-DA mute |
| CdlDemute | 0x0c | B | Release mute |
| CdlSetfilter | 0x0d | B | Select play ADPCM |
| CdlSetmode | 0x0e | B | Set basic mode |
| CdlGetlocL | 0x10 | B | Get logical location (data sector) |
| CdlGetlocP | 0x11 | B | Get physical location (audio sector) |
| CdlSeekL | 0x15 | N | Logical seek (data sector seek) |
| CdlSeekP | 0x16 | N | Physical seek (audio sector seek) |
| CdlReadS | 0x1b | B | Start data read (no retry) |
| CdlReset | 0x1c | B | Reset |

B: Blocking; N: Non-blocking operation

### Return Value

Last command

### Remarks

**See also:**

## CdLastPos

Obtains the CD-ROM location most recently specified.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

**Syntax**

**CdlLOC *CdLastPos** (**void**)

**Arguments**

None.

**Explanation**

This function returns the latest location that was specified by the sub command, CdlSetloc/CdlPlay/CdlSeekL/CdlSeekP/CdlRead/CdlReadS.

**Return value**

Pointer to the structure, CdlLOC containing the CD-ROM location.

**Remarks**

**See also:**

## CdMix

Set attenuation for CD audio.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdMix** (**CdlATV** **vol*)
**CdlATV** **vol*;

### Arguments

| | |
|---|---|
| *vol* | Pointer to attenuator volume |

### Explanation

Set audio volume value for CD audio (CD-DA, ADPCM).

### Return value

Always returns 1.

### Remarks

**See also:**

## CdMode

Obtains the latest CD-ROM mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**int CdMode** (**void**)

### Arguments

None.

### Explanation

This function returns the latest CD-ROM mode set.

### Return value

CD-ROM mode.

### Remarks

High speed since this function only refers to the status in the main memory. Status buffer is updated when a CD-ROM command is issued. It is required to issue CdlNop command in order to obtain the latest state explicitly.

**See also:**

## CdPlay

Plays CD-DA tracks.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**int CdPlay** *(***int** *mode*, **int** **tracks*, **int** *offset)*

### Arguments

| | |
|---|---|
| *mode* | 0: Stops playing |
| | 1: Plays track numbers specified in the *tracks* array in the specified order. Stop at end. |
| | 2: Plays track numbers specified in the *tracks* array in the specified order. Repeat at end. |
| | 3: Returns an index of the array corresponding to the track currently being played. |
| *tracks* | Pointer to array specifying the track to be played. Must ends with 0. |
| *offset* | Index of the "tracks" to be played. |

### Explanation

This function plays multiple tracks specified by the array "tracks" in order. After playing the last track of the array, it repeats or stops playing according to the mode specified.

### Return value

Index of the "tracks" currently being played. Not the track number. -1 when it has already stopped playing.

### Remarks

All playing is done in the unit of track. The playing or stopping in the middle of the track is not allowed.

**See also:**

## CdPosToInt

Translate CD position information from a minutes/seconds/sector time code to an absolute sector number.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdPosToInt** (**p*)
**CdlLOC** **p*;

### Arguments

| | |
|---|---|
| *p* | Pointer to a cdlLOC structure that contains the position time code. |

### Explanation

Translate a minutes/seconds/sectors time code contained in a cldLOC structure pointed to by the *p* parameter into an absolute sector value.

### Return value

Absolute sector number.

### Remarks

**See also:**

## CdRead

Read multiple sectors from the CD-ROM.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdRead** (*sectors*, **buf*, *mode*)
**int** *sectors*;
**unsigned long** **buf*;
**int** *mode*;

### Arguments

| | |
|---|---|
| *sectors* | Read sector count |
| *buf* | Pointer to read buffer |
| *mode* | CD-ROM subsystem mode, as defined for CdlSetMode command (see the description of the CdControl() function). |

### Explanation

Reads one or more sectors of data from the CD-ROM to the specified buffer in memory. The starting position for the read is the position last specified for CdlSeekL or CdlSetloc, or the next sector following the previous CdRead() call.

The CdRead() call is non-blocking. Check for completion using the CdReadSync() or CdReadCallback() functions. The CdRead() function uses the CdReadyCallback() function internally, so that function cannot be used with CdRead().

### Return value

1 if command issued successfully, otherwise 0.

### Remarks

Note that the return code from CdRead only indicates if the command was issued successfully or not. For information about CD-ROM errors which occur during reading, check the result array of the CdReadSync function.

**See also:**

## CdRead2

Starts reading data from the CD-ROM.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdRead2** (*mode*)
**int** *mode*;

### Arguments

| | |
|---|---|
| *mode* | CD-ROM subsystem mode, as defined for CdlSetMode command (see the description of the CdControl() function). |

### Explanation

Seeks to the position specified by CdlSetloc and commences reading data into the internal sector buffer. Commences streaming when the CdlModeStream flag is set in the *mode* parameter. Commences ADPCM audio play when the CdlModeRT flag is set in the *mode* parameter.

This function must be used in conjunction with the CdGetSector() function to transfer data from the internal sector buffer to the program's desired destination buffer. The CdGetSector() function should be called to transfer data as soon as either the CdReady() or CdReadyCallback() functions return the CdlDataReady flag.

### Return value

1 if command issued successfully, otherwise 0.

### Remarks

**See also:**

## CdReadCallback

Defines a callback function to be executed on completion of CdRead.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long CdReadCallback** (**func*)
**void** (**func*)(**int** *status*, **unsigned char** **result*);

### Arguments

| | |
|---|---|
| *func* | Pointer to callback function address |
| *status* | Return code of the CdReadSync() |
| *result* | Pointer to an 8-byte array containing status and result information |

### Explanation

*func* defines the callback called when CdRead() completes. *func* is passed two arguments. The *status* argument will be either CdlComplete or CdlDiskError, corresponding to the return code of the CdReadSync() function. The *result* argument is a pointer to an 8-byte array containing status and result information, corresponding to the *result* argument of the CdReadSync() function.

If *func* is set as 0, callback does not occur.

### Return value

Address of previously set callback

### Remarks

While *func* is executing, subsequent data transfer complete interrupts are masked. Therefore, *func* should return as soon as the necessary processing is completed.

To restore the previous callback, preserve the return value and when processing finishes, use it to restore the previous callback address.

**See also:**

## CdReadExec

Loads PlayStation-format executable program file from CD-ROM.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**struct EXEC *CdReadExec** (**char** **file*)

### Arguments

| | |
|---|---|
| *file* | Pointer to executable file name |

### Explanation

This function loads the executable program specified by *file* into main memory at the address specified by the program file header. The file must be an executable program in the PlayStation EXE format. To determine when the load is complete, use the CdReadSync() or CdReadCallback() functions. After loading, the program can be executed as a child process using the Exec() function.

### Return value

Pointer to an EXEC structure that describes the loaded program.

### Remarks

Load address of the executable file should not overlap with the region of its parent process.

**See also:**

## CdReadFile

Reads a file on CD-ROM.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**int CdReadFile** (**char** **file*, **u_long** **addr*, **int** *nbyte*)

### Arguments

| | |
|---|---|
| *file* | Pointer to file name |
| *addr* | Pointer to main memory address to be read-in |
| *nbyte* | Data size to be read-in |

### Explanation

This function reads *nbyte* bytes of data from the specified file on the CD_ROM. If *nbyte* is zero, the entire file is read. If *file* is NULL, the function starts reading from the last location of the previous CdReadFile call.

### Return value

| | |
|---|---|
| 0 | Read error |
| Other | Number of bytes read |

### Remarks

The filename must contain a full path specification. All lowercase letters will be converted to uppercase. Reading is performed in the background. Use CdReadSync() or CdReadCallback() to determine when reading is completed.

**See also:**

## CdReadSync

Wait for completion of CdRead and related CD-ROM functions.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdReadSync** (*mode*, **result*)
**int** *mode*;
**unsigned char** **result*;

### Arguments

| | |
|---|---|
| *mode* | Await read completion. |
| *result* | Pointer to status storage buffer of command most recently completed. |

### Explanation

Checks the current status of a data read operation initiated by CdRead, CdReadFile, and other related functions. Depending on the value of the *mode* parameter, either returns the current status immediately or waits for the operation to complete.

**Table 9–7**

| Value | Contents |
|---|---|
| 0 | Waits for completing of read and returns |
| 1 | Determines current status and promptly returns |

### Return value

Returns the values below.

**Table 9–8**

| Return value | Contents |
|---|---|
| Positive integer | Number of sectors remaining |
| 0 | Completion |
| -1 | Read error |

### Remarks

**See also:**

## CdReady

Wait for CD-ROM data to be ready.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdReady** (*mode*, **result*)
**int** *mode*;
**unsigned char** **result*;

### Arguments

| | |
|---|---|
| *mode* | Wait until data is prepared. |
| *result* | Pointer to status storage buffer of command most recently completed. |

### Explanation

This function is used after a CD-ROM read is initiated using CdRead2(), CdControl (CdlReadS), or CdControl (CdlReadN) to determine if there is data available in the sector buffer which is ready to be transferred using the CdGetSector() function. Depending on the value of the *mode* parameter, either returns the current status immediately or waits for the operation to complete.

**Table 9–9**

| Value | Contents |
|---|---|
| 0 | Data waits until it can be prepared and returns |
| 1 | Determines current status and promptly returns |

### Return value

Data-available status is indicated by the following values:

**Table 9–10**

| Return value | Meaning |
|---|---|
| CdlDataReady | There is data available for transfer |
| CdlDiskError | Error detected |
| CdlNoIntr | No preparation-completed data |

### Remarks

**See also:**

## CdReadyCallback

Define CdReady callback function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long CdReadyCallback** (**func*)
**void** (**func*)(int *status*, unsigned char **result*);

### Arguments

| | |
|---|---|
| *func* | Pointer to callback function address |
| *status* | Return code of the CdReadySync() |
| *result* | Pointer to an 8-byte array containing status and result information |

### Explanation

Defines a callback routine to be executed when there is data available in the sector buffer following a CD-ROM read initiated using CdRead2(), CdControl (CdlReadS) or CdControl (CdlReadN). The *func* parameter specifies the address of the desired callback routine. If *func* is NULL, any previous callback routine is disabled.

*func* is passed two arguments. The *status* argument will be either CdlComplete or CdlDiskError, corresponding to the return code of the CdReadySync() function. The *result* argument is a pointer to an 8-byte array containing status and result information, corresponding to the *result* argument of the CdReadySync() function.

### Return value

Address of previously set callback.

### Remarks

While *func* is executing, subsequent data available interrupts are masked. Therefore *func* should return as soon as the necessary processing is completed.

To restore the previous callback, preserve the return value and when processing finishes, use it to restroe the previous callback address.

**See also:**

## CdReset

Initialization of CD-ROM subsystem.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdReset (**
**int** *mode*
**)**

### Arguments

| | |
|---|---|
| *mode* | Reset mode |

### Explanation

Initializes the CD-ROM subsystem. CdInit () low-level function.

Unlike CdInit, this function does not initialize the event environment related to CD-ROM.

In reset mode the following values can be specified

**Table 9–11**

| Mode | Contents |
|---|---|
| 0 | Initialization of CD subsystem only |
| 1 | Initialization of CD subsystem and CD audio volume (CD-DA, ADPCM) |

When mode has been specified as 0, and initialization of CD audio volume is not performed, the volume settings specified in previous sound libraries will be saved.

### Return value

If initialization is successful, returns 1. If fails, returns 0.

### Remarks

No retry is carried out.

**See also:** CdInit (p. 9-21).

## CdSearchFile

Get location and size from CD-ROM file name.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**CdlFILE *CdSearchFile** (**fp*, **name*)
**CdlFILE** **fp*;
**char** **name*;

### Arguments

| | |
|---|---|
| *fp* | Pointer to CD-ROM file structure pointer |
| *name* | Pointer to a file name |

### Explanation

Determine the position time code (minutes, seconds, sectors) and total length of the specified file on the CD-ROM. The result is stored in the CdlFILE structure pointed to by the *fp* parameter.

### Return value

Returns 0 on failure. On success returns a pointer to the *fp* structure.

### Remarks

The *file* specification must be a complete path to the file.

The CdSearchFile() function caches directory information, so subsequent consecutive calls for files in the same directory do not require additional CD-ROM reads. Note that only one directory is cached at a time and reading information for a file in another directory will invalidate the entire cache.

For the best possible performance, include file location and size information in your program at compile time instead of using CdSearchFile.

**See also:**

## CdSetDebug

Set debug level.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdSetDebug** (*level*)
**int** *level*;

### Arguments

| | |
|---|---|
| *level* | Debug level |

### Explanation

Set debug level for CD-ROM subsystem. The possible values of *level* are shown below.

**Table 9–12**

| Value | Contents |
|---|---|
| 0 | No checks performed |
| 1 | Check primitive commands |
| 2 | Print execution status of primitive commands |

### Return value

Previously set debug mode.

### Remarks

**See also:**

## CdStatus

Obtains the latest CD-ROM status.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.0* | *7/31/96* |

### Syntax

**int CdStatus** (**void**)

### Arguments

None.

### Explanation

This function obtains the latest reported CD-ROM status.

### Return value

CD-ROM Status.

### Remarks

This function operates at high speed because it simply returns the status code maintained by the CD-ROM system. The status buffer is updated whenever a CD-ROM command is issued. To explicitly obtain the absolute most current status, issue a CdControl(CdlNop) command immediately before your CdStatus() call.

**See also:**

## CdSync

Wait for completion of CD-ROM command.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int CdSync** (*mode*, **result*)
**int** *mode*;
**unsigned char** **result*;

### Arguments

| | |
|---|---|
| *mode* | Waits for command termination |
| *result* | Pointer to status storage buffer of command most recently completed. |

### Explanation

Waits for actual termination of a command issued by CdControl(). The *mode* parameter specifies whether to wait and return command termination.

**Table 9–13**

| Value | Contents |
|---|---|
| 0 | Waits for command termination and returns |
| 1 | Determines current status and promptly returns |

### Return value

Command execution status is indicated by the following values:

**Table 9–14**

| Return value | Meaning |
|---|---|
| CdlComplete | Command complete |
| CdlDiskError | Error detected |
| CdlNoIntr | Command is being executed |

### Remarks

**See also:**

## CdSyncCallback

Define CdSync callback function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long CdSyncCallback** (**func*)
**void** (**func*)(**int** *status*, **unsigned char** **result*);

### Arguments

| | |
|---|---|
| *func* | Callback function address |
| *status* | Return code of CdSync() function |
| *result* | Pointer to an 8-byte array containing status and result information |

### Explanation

Defines a callback routine to be executed when a CdControl() command is completed. The *func* parameter specifies the address of the desired callback routine. If *func* is NULL, any previous callback routine is disabled.

*func* is passed two arguments. The *status* argument will be either CdlComplete or CdlDiskError, corresponding to the return code of the CdSync() function. The *result* argument is a pointer to an 8-byte array containing status and result information, corresponding to the *result* argument of the CdSync() function.

### Return value

Address of previously set callback.

### Remarks

While *func* is executing, subsequent CD-ROM command complete interrupts are masked. Therefore, *func* should return as soon as the necessary processing is completed.

To restore the previous callback, preserve the return value and when processing finishes, use it to restore the previous callback address.

**See also:**

## StCdInterrupt

Handler for interrupts from CD-ROM (internal function).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void StCdInterrupt** (*void*)

### Arguments

None.

### Explanation

This function is normally hooked to CD-ROM interrupts by StStartStream() and StStartEmulation(), and it is called automatically at interrupt generation, so it does not need to be called by the user. When used in 24-bit mode, the interrupt just sets StCdInterFlag, so this function needs to be called by the application.

### Return value

None.

### Remarks

**See also:**

## StClearRing

Flush ring buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void StClearRing** (*void*)

### Arguments

None.

### Explanation

Flush ring buffer. Flushing the ring buffer when jumping tracks and so forth is effective in preventing excess frames from showing up.

### Return value

None.

### Remarks

**See also:**

## StFreeRing

Release ring buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long StFreeRing** (**base*)
**unsigned long** **base*;

### Arguments

| | |
|---|---|
| *base* | Pointer to starting address of user data area of released 1 frame |

### Explanation

The area obtained by StGetNext() is locked. StFreeRing() releases this locked region. The released region is the region for one frame's worth of data which is used as the base for the starting address of the user region. Linked sector header regions are also released.

If a region locked by StGetNext() is not released when its use ends, the ring buffer will soon overflow and streaming will come to a halt.

### Return value

A return value of 0 indicates successful release. 1 denotes a failed release (for example, trying to release something that wasn't locked).

### Remarks

**See also:**

## StGetBackloc

Returns the location and ID of the first frame in the ring buffer in order to access frame data without any frame skip. The frame skip due to ring buffer overflow can be avoided by re-accessing the frame location obtained by this function. This function is not appropriate for data with XA AUIO since it requires data access.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**int StGetBackloc** *(*loc)*
**CdlLOC** **loc;*

### Arguments

*loc* Pointer to latest location of the first frame.

### Explanation

This function returns the latest location information and ID of the frame on the current ring buffer.

The location information obtained here is used as the access target value in order to avoid frame skip due to ring buffer overflow.

Please refer to \psx\sample\cd\movie\tuto3.c for usage example.

This function is valid only for StModeStream2 mode.

### Return value

Frame ID that should be used upon the streaming restart. -1 for error indicating non StModeStream2 mode.

### Remarks

**See also:**

## StGetNext

Get one frame of ring buffer data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**unsigned long StGetNext** (**addr*, **header*)
**unsigned long** **addr*;
**unsigned long** **header*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to user data region starting address for 1 frame of retrieved data |
| *header* | Pointer to sector header region starting address for 1 frame of retrieved data |

### Explanation

This function gets one frame of ring buffer data. If the next frame of data is ready in the ring buffer, the starting address of the user data and the sector header are stored in *addr* and *header* respectively. 0 is returned.

The region the data is taken from is locked until StFreeRing() is called, so it cannot be destroyed by new data.

The data region has a continuous address and the ring buffer does not loop in mid-data.

### Return value

If 1 FRAME of data is taken from the ring buffer, 0 is returned. If it is not ready, 1 is returned.

### Remarks

**See also:**

## StGetNextS

Get one frame of ring buffer data from memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**unsigned long StGetNext** (**addr*, **header*)
**unsigned long** **addr*;
**unsigned long** **header*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to user data region starting address for 1 frame of retrieved data |
| *header* | Pointer to sector header region starting address for 1 frame of retrieved data |

### Explanation

This function gets one frame of ring buffer data. The starting addresses and the sector header are stored in *addr* and *header* respectively. 0 is returned.

### Return value

When one frame of data is taken from the ring buffer, 0 is returned.

### Remarks

**See also:**

## StNextStatus

Returns the status of the next frame. This function checks whether the next frame data is available on the ring buffer without affecting the internal state.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**unsigned long StNextStatus** (**addr*, **header*)
**unsigned long** **addr*;
**unsigned long** **header*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to starting address of the user data region for 1 frame of retrieved data |
| *header* | Pointer to starting address of sector header region for 1 frame of retrieved data |

### Explanation

This function obtains the status of the next frame of ring buffer data.

The internal state is not affected by calling this function.

Following is the possible status:

| | |
|---|---|
| StFREE | Next frame is not on the ring buffer. |
| StCOMPLETE | Next frame is completely read into the ring buffer. |
| StBUSY | Next frame is being read into the ring buffer. |
| StLOCK | Next frame is being processed (one frame is obtained by calling StGetNext but StFreeRing has not been called). |

### Return value

Next frame status as shown above.

### Remarks

**See also:**

## StRingStatus

Returns the status of the ring buffer. Frame skip caused by insufficient free space in the ring buffer can be prevented by calling this function.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *3.5* | *7/31/96* |

### Syntax

**void StRingStatus** (**free_sectors*, **over_sectors*)
**short** **free_sectors*,**over_sectors;*

### Arguments

| | |
|---|---|
| *free_sectors* | Pointer to the number of free sectors on the ring buffer. |
| *over_sectors* | Pointer to the difference between the sector positions of CD-ROM data read in and the sector positions currently being processed. |

### Explanation

This function reports the ring buffer status with two variables specified as arguments.

The first argument, "free_sectors," is the number of sectors with no data in the unused area of the ring buffer. The larger the "free_sectors" is, the more free space on the ring buffer.

The second argument, "over_sectors," is the difference between the sector positions for CD-ROM data read in and the sector positions currently being processed. The larger the "over_sectors" is, the more unprocessed data on the ring buffer.

The sum of "free_sectors" and "over_sectors" and the total ring buffer size is nearly equal. The reason for not having an exact match in size is that when one frame cannot fit in completely close to the end, rewind occurs.

### Return value

None.

### Remarks

**See also:**

## StSetChannel

Set streaming channel.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**int StSetChannel** (*ch*)
**unsigned long** *ch*

### Arguments

| | |
|---|---|
| *ch* | Playback channel |

### Explanation

Sets streaming playback channel. *ch* sets the channel (0-31). The channel stores the STR data at the authoring level.

### Return value

If the channel is set, return 0; otherwise, return 1.

### Remarks

**See also:**

## StSetEmulate

Set parameters for streaming emulation.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void StStartEmulate**(**addr*, *loc*, *start_frame*, *end_frame*, *f1*, *f2*)
**unsigned long** **addr*, *loc*, *start_frame*, *end_frame*;
**int** (**func1*)(), (**func2*)();

### Arguments

| | |
|---|---|
| *addr* | Pointer to emulation data starting address |
| *loc* | Set color mode |
| *start_frame* | Streaming start frame |
| *end_frame* | Streaming end frame |
| *func1* | Address of function called back for each frame of data |
| *func2* | Address of function called back when streaming ends |

### Explanation

Sets parameters for streaming emulation. Emulation means that CD-ROM data is put into memory in advance and data streaming is performed from memory, not from the CD-ROM, which provides only data-ready timing. In streaming emulation, play time is limited to a few seconds because of limits in memory capacity. Still, emulation is easier than using a CD-ROM emulator.

STR-format data needs to be loaded to *addr* in advance. See StSetStream() for details on other arguments. (*loc* is the same as *mode*.)

### Return value

None.

### Remarks

**See also:**

## StSetMask

Controls the playing of streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void StSetMask** (*mask*, *start*, *end*)
**unsigned long** *mask*, *start*, *end*;

### Arguments

| | |
|---|---|
| *mask* | Streaming play on/off |
| *start* | StSetStream() start_frame |
| *end* | StSetStream() end_frame |

### Explanation

Turns streaming play ON/OFF. There is no mechanical timing lag compared to CD-ROM drive pause and playback, and instant ON/OFF is possible.

Values that can be specified in *mask* are as follows.

**Table 9–15**

| Value | Contents |
|---|---|
| 0 | Play |
| 1 | Pause |

Resets start and end of SetStream() trigger frame values.

### Return value

None.

### Remarks

**See also:**

## StSetRing

Set ring buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void StSetRing** (**ring_addr*, *ring_size*)
**unsigned long** **ring_addr*;
**unsigned long** *ring_size*;

### Arguments

| | |
|---|---|
| *ring_addr* | Pointer to ring buffer starting address |
| *ring_size* | Ring buffer size (in sectors) |

### Explanation

Secure a ring buffer of a size specified by *ring_size* from an address specified by *ring_addr*.

To use the Streaming Library, you must first call it.

Because only form-1 CD-ROM sectors are supported at present, one sector of data area is 2048 bytes.

It is necessary to secure this area in the main program.

### Return value

None.

### Remarks

**See also:**

## StSetStream

Set streaming parameters.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void StSetStream** (*mode*, *start_frame*, *end_frame*, *f1*, *f2*)
**unsigned long** *mode*, *start_frame*, *end_frame*;
**int** (**func1*)(), (**func2*)();

### Arguments

| | |
|---|---|
| *mode* | Set color mode |
| *start_frame* | Frame to start streaming |
| *end_frame* | Frame to end streaming |
| *func1:* | Address of function called back for each 1 FRAME of data. |
| *func2:* | Address of function called back when streaming ends. |

### Explanation

Sets streaming parameters.

The specified values and contents of each argument are as follows:

a) *mode*

Sets color mode. The values you may specify are as follows:

**Table 9–16**

| Value | Contents |
|---|---|
| 0 | 16-bit mode |
| 1 | 24-bit mode |

b) *start_frame*

Specifies the frame number (stored in STR data) that starts streaming.

Streaming will not begin until this Streaming Library frame is reached. If you want to play the data starting in the middle, you must specify an appropriate frame number. When you specify 0, streaming commences no matter what the frame number is.

c) *end_frame*

Specifies the frame number (stored in STR data) that ends streaming. Streaming ends when this Streaming Library frame is reached. If you specify a number large enough, it plays the CD-ROM data to the end and termintes. When you specify 0, all the data is stored in the ring buffer and the function automatically terminates. This takes a large ring buffer, and the function is successful when streaming is from memory.

d) *func1*

Generates one frame's worth of data and specifies the address of the callback function called.

e) *func2*

Sets the address of the callback function called at the time streaming is completed.

### Return value

None.

### Remarks

To correctly exit from a streaming application, the end of streaming should not be set by end_frame. Set end_frame to 0xffffffff, and code an appropriate endpoint from within the loop.

### See also:

## StUnSetRing

Release interrupt used by streaming library.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcd.lib* | *Libcd.h* | *2.x* | *7/31/96* |

### Syntax

**void StUnSetRing** (*void*)

### Arguments

None.

### Explanation

Release two interrupt functions CdDataCallback() and CdReadyCallback() hooked by CDRead2(CdlModeStream) and return to initial state.

If the streaming library is not used when streaming ends and control transfers to another program, the interrupt hooks which call this function need to be returned to the initial state.

### Return value

None.

### Remarks

**See also:**

# Chapter 10: Controller/Peripherals Library
## Table of Contents

## InitGUN

Initializes light gun driver.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgun.lib* | *Libgun.h* | *3.5* | *11/1/96* |

### Syntax

**void InitGun (char** **bufA*, **long** *lenA*, **char** **bufB, long lenB, char *buf1, char *buf2, long len* **)**

### Arguments

| | |
|---|---|
| *bufA* | Pointer to buffer that will receive standard controller information for port 1. |
| *lenA* | Length of *bufA.* Must be 34 bytes long to support both directly connected controllers and Multi Tap |
| *bufB* | Pointer to buffer that will receive standard controller information for port 2. |
| *lenB* | Length of *bufA.* Must be 34 bytes long to support both directly connected controllers and Multi Tap |
| *buf1* | Pointer to receive light gun position data for port 1. |
| *buf2* | Pointer to receive light gun position data for port 2. |
| *len* | Length of *buf1* and *buf2*, specified in long words. (20 maximum) |

### Explanation

When the light gun detects the position of the television's electron beam as the current screen is drawn, the vertical scanline counter and horizontal pixel clock values are stored into the specified buffers. The status of the gun trigger and other buttons is also returned, but does not affect the position reading (i.e. the position data is returned regardless of the button status).

The buffers specified by *bufA* and *bufB* are used to receive controller data from non-light gun controllers, as well as the button information from the light gun. These buffers are identical to those specified by the InitPAD and InitTAP functions. See the documentation for those functions for further details.

The Gun_Buffer data structure defined below shows the format of the buffers specified by *buf1* and *buf2.* It contains a *status* value, and a *count* value representing how many coordinate pairs were captured during the previous video screen. The coordinate pairs themselves are contained in the *gunpos* array. The *len* parameter to the InitGUN function specifies the number of elements in the *gunpos* array.

```
typedef struct
{
    unsigned short      v_count;
    unsigned short      h_count;
} Gun_Position;

typedef struct
{
    unsigned char status;
    unsigned char count;
    Gun_Position  gunpos[20];
} Gun_Buffer;
```

The *y_count* field of the Gun_Position structure will contain the vertical position of the gun expressed as the number of vertical scanlines since the previous vertical blanking period. The vertical offset of your display screen should be subtracted from this value to obtain the actual pixel position. The *h_count* field contains the horizontal position of the gun. This is expressed as the number of ticks of the horizontal pixel clock since the beginning of the current scanline and must also be converted to a pixel value. See the sample light gun programs supplied by Sony for examples of converting these values.

The light gun driver is capable of capturing as many as 20 sets of gun position coordinates per screen, but more typically somewhere from 2 to 4 sets of coordinates will be captured.

## Return value

None.

## Remarks

InitGUN combines the earlier InitGun and separate InitPAD functions and is now fully compatible with the Multi Tap.

Because InitGUN supports the Multi Tap multi-controller adapter, the size of the buffers *bufA* and *bufB* must be large enough to receive all of the data (i.e. 34 bytes). Otherwise your program will not function properly with the Multi Tap adapter.

The light gun must generate an interrupt to signal where the gun is aimed, so aside from InitGUN, additional hardware events must be initialized. See the controller sample program supplied by Sony for examples of proper initialization.

**See also:** SelectGun (p. 10-12). InitPAD (p. 10-5), InitTAP (p. 10-6), StartGUN (p. 10-13), StopGUN (p. 10-16), RemoveGUN (p. 10-11)

## InitPAD

Initializes the controller.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long InitPAD** (**bufA*, *lenA*, **bufB*, *lenB*)
**char** **bufA*, **bufB*;
**long** *lenA*, *lenB*;

### Arguments

| | |
|---|---|
| *bufA*, *bufB* | Pointers to incoming data buffers |
| *lenA*, *lenB* | Length of incoming data buffers (in bytes) |

### Explanation

This function registers a receive data buffer for the controller. The format of received data is given in the Library Overview. ChangeClearPAD() is not executed internally.

### Return value

Always 1.

### Remarks

**See also:** StartPAD (p. 10-14), StopPAD (p. 10-17), ChangeClearPAD (p. 1-11).

## InitTAP

Controller initialization, including Multi Tap support

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libtap.lib* | *Libtap.h* | *3.4* | *1/11/96* |

### Syntax

**void InitTAP ( char** **bufA*, **long** *lenA*, char **bufB*, **long** *lenB* **)**

### Arguments

| | |
|---|---|
| *bufA* | Pointer to data buffer that will receive controller data for port 1. |
| *lenA* | Length in bytes of buffer specified by *bufA*. Should be 34 bytes to support Multi Tap. |
| *bufB* | Pointer to data buffer that will receive controller data for port 2. |
| *lenB* | Length in bytes of buffer specified by *bufA*. Should be 34 bytes to support Multi Tap. |

### Explanation

This function installs a routine that reads the controllers during the vertical blank period and stores the controller data into the specified buffers. This routine differs from the one installed by InitPAD mainly in that it supports a larger data buffer for the Multi Tap multi-controller adapter, and also that 8 bytes of data are always transferred for each controller, even if some are unused.

### Return value

None.

### Remarks

Use InitTAP instead of InitPAD when Multi Tap support is required. Use InitGUN instead of InitTAP when light gun support is also required.

The 34 byte buffer returned by the Multi Tap is organized as shown below.

**Table 10–1: Multi-tap buffer configuration**

| Byte(s) | Contents |
|---|---|
| 0 | Received result<br>0x00: success<br>0xFF: failure |
| 1 | Multi Tap ID code: 0x80 |
| 2-9 | Data for controller connected to Multi Tap port A |
| 10-17 | Data for controller connected to Multi Tap port B |
| 18-25 | Data for controller connected to Multi Tap port C |
| 26-33 | Data for controller connected to Multi Tap port D |

The data stored for each controller port is as shown below. Note that not all controller types will return useful information in all fields.

**Table 10–2: Generic 8-byte Controller Data Buffer Format**

| Byte | Contents |
|---|---|
| 0 | Received result<br>0x00: success<br>0xFF: failure |
| 1 | High order 4-bits: controller classification |

| | |
|---|---|
| | 0x1: Mouse<br>0x2: 16 button analog joystick controller<br>0x3: Gun controller<br>0x4: 16 button digital controller<br>0x5: Dual analog joystick controller<br>0x8: Multi-tap multi-controller adapter<br>Low-order 4 bits: (# of data bytes following / 2) |
| 2 | Digital button conditions; 1: released, 0: pushed<br>Bit 7 = D-pad Left<br>Bit 6 = D-pad Down<br>Bit 5 = D-pad Right<br>Bit 4 = D-pad Up<br>Bit 3 = Start Button<br>Bit 2 = not used<br>Bit 1 = not used<br>Bit 0 = Select Button |
| 3 | Digital button conditions; 1: released, 0: pushed<br>Bit 7 = □ Button<br>Bit 6 = ✕ Button<br>Bit 5 = Δ Button<br>Bit 4 = ○ Button<br>Bit 3 = L1 Button<br>Bit 2 = R1 Button<br>Bit 1 = L2 Button<br>Bit 0 = R2 Button |
| 4 | Analog channel A (value interpreted as either -128 to 127, or 0-255) |
| 5 | Analog channel B (value interpreted as either -128 to 127, or 0-255) |
| 6 | Analog channel C (value interpreted as either -128 to 127, or 0-255) |
| 7 | Analog channel D (value interpreted as either -128 to 127, or 0-255) |

**See also:** StartTAP (p. 10-15), StopTAP (p. 10-18).

## PadInit

Initializes a controller.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *3.0* | *7/31/96* |

### Syntax

**void PadInit** (*mode*)

### Arguments

| | |
|---|---|
| *mode* | Controller type |

### Explanation

This function initializes all connected controllers of the type specified by the *mode* parameter

### Return value

None.

### Remarks

At present, only type 0 controllers are supported. For more general and comprehensive controller support, use the InitPAD function instead.

**See also:**

## PadRead

Read data from the controller.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long PadRead** (*id*)
**unsigned short** *id*;

### Argument

| | |
|---|---|
| *id* | Controller ID |

### Explanation

This function reads data from the controller specified by the *id* parameter.

### Return value

The return value is the controller button status, corresponding to bytes 2 & 3 as shown in Table 10–2.

### Remarks

Currently, the *id* parameter has no meaning.

**See also:**

## PadStop

Halts controller.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libetc.lib* | *Libetc.h* | *2.x* | *7/31/96* |

### Syntax

**void PadStop** *( void )*

### Argument

None.

### Explanation

Halts all currently connected controllers.

### Return value

None.

### Remarks

When processing is complete, it is necessary to call this function without fail and halt the controller driver.

**See also:**

## RemoveGUN

Disables interrupts used by light gun driver.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgun.lib* | *Libgun.h* | *3.6* | *11/1/96* |

### Syntax

**void RemoveGun (void)**

### Arguments

*none*

### Explanation

Disables and removes the interrupt routines used by the light gun driver.

### Return value

None.

### Remarks

Use RemoveGUN prior to overwriting the memory used by your program (such as when an overlay containing the gun code is being replaced, or prior to a Load/Exec sequence). Failure to do so will likely result in a program crash.

The RemoveGUN function replaces the ResetGun function from library v3.5.

**See also:** StopGUN (p. 10-16), InitGUN (p. 10-3)

## SelectGUN

Selects gun.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgun.lib* | *Libgun.h* | *3.6* | *1/11/96* |

### Syntax

**void SelectGun** (**int** *ch*, **unsigned char** *mask*)

### Arguments

| | |
|---|---|
| *ch* | Gun channel (0 or 1) |
| *mask* | Interruption mask setting<br>0: interruption prohibited<br>1: interruption permitted) |

### Explanation

Reports the on/off of the interruption mask for the gun.

It is not possible to cancel more than two masks at the same time.

### Return value

### Remarks

**See also:** InitGun (p. 10-3).

## StartGUN

Starts controller reading.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgun.lib* | *Libgun.h* | *3.6* | *11/4/96* |

### Syntax

**long StartGUN** ( void )

### Arguments

None.

### Explanation

Allows the light gun position reading initiated by InitGUN to store data into the light gun controller data buffers.

### Return value

When successful, returns 1. When fails, returns 0

### Remarks

The default state after InitGUN is the same as after StartGUN.

The StartGUN routine replaces the StartGun routine available in library v3.5.

**See also:** InitGUN (p. 10-3), StopGUN (p. 10-16)

## StartPAD

Starts reading the controller.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**long StartPAD** (*void*)

### Arguments

None.

### Explanation

Triggered by the interruption of a vertical retrace line, this function starts to read the controller. ChangeClearPAD (1) is executed internally.

### Return value

Always returns 1.

### Remarks

Interruption is permitted.

**See also:** InitPAD (p. 10-5), ChangeClearPAD (p.1-11).

## StartTAP

Starts controller reading.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libtap.lib* | *Libtap.h* | *3.4* | *11/1/96* |

### Syntax

**long StartTAP** ( void )

### Arguments

None.

### Explanation

Allows the controller reading routing installed by InitTAP to store data into the controller data buffers.

### Return value

When successful, returns 1. When fails, returns 0

### Remarks

The default state after InitTAP is the same as after StartTAP.

**See also:** InitTAP (p. 10-6).

## StopGUN

Halts controller reading.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libgun.lib* | *Libgun.h* | *3.6* | *11/1/96* |

### Syntax

**void StopGUN** ( void )

### Arguments

None.

### Explanation

Disables the controller reading routine from storing controller data into the buffers installed with InitGUN. The controller reading interrupts still occur, but the data is not stored.

### Return value

None.

### Remarks

Use StopGUN in place of StopPAD or StopTAP when light gun support is desired.

**See also:** InitTAP (p. 10-6), StopPAD (p. 10-17), StopTAP (p. 10-18)

## StopPAD

Stops reading the controller.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libapi.lib* | *Kernal.h* | *2.x* | *7/31/96* |

### Syntax

**void StopPAD** (*void*)

### Arguments

None.

### Explanation

This function stops reading the controller. Interruption is not permitted.

### Return value

None.

### Remarks

**See also:** InitPAD (p. 10-5), ChangeClearPAD (p. 1-11).

## StopTAP

Halts controller reading.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libtap.lib* | *Libtap.h* | *3.4* | *11/1/96* |

### Syntax

**void StopTAP** ( void )

### Arguments

None.

### Explanation

Disables the controller reading routine from storing controller data into the buffer installed with InitTAP. The controller reading interrupt still occurs, but the data is not stored.

### Return value

None.

### Remarks

Use StopTAP in place of StopPAD when Multi Tap support is desired. Use StopGUN when light gun support is also required.

**See also:** InitTAP (p. 10-6), StopPAD (p. 10-17), StopGUN (p. 10-16)

# Chapter 11: Link Cable Library
Table of Contents

## AddCOMB

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcomb.lib* | *Libcomb.h* | *3.0* | *7/31/96* |

### Syntax

**AddCOMB** (*void*)

### Arguments

None.

### Explanation

Initialize link cable driver.

### Return values

### Remarks

**See also:**

## ChangeClearSIO

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcomb.lib* | *Libcomb.h* | *3.0* | *7/31/96* |

### Syntax

**ChangeClearSIO** (*val*)
**Long** *val*

### Arguments

| | |
|---|---|
| *val* | Interrupt cause clear flag |

### Explanation

If *val* is set as non-0, an interrupt from an expansion SIO in the driver is cleared. This is used only when other expansion SIO drivers are also present.

### Return values

### Remarks

**See also:**

## DelCOMB

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcomb.lib* | *Libcomb.h* | *3.0* | *7/31/96* |

### Syntax

**DelCOMB** (*void*)

### Arguments

None.

### Explanation

Remove link cable driver from kernel.

### Return values

### Remarks

**See also:**

## _comb_control

Combat cable BIOS.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libcomb.lib* | *Libcomb.h* | *3.0* | *7/31/96* |

### Syntax

**long _comb_control** (*cmd*, *arg*)
**long** *cmd*;
**long** *arg*;

### Arguments

| | |
|---|---|
| *cmd* | Control command |
| *arg* | Control command argument |

### Explanation

Offers the same functionality as ioctl() to an sio device.

### Return value

The return value depends on the control command used in cmd.

### Remarks

**See also:**

# Chapter 12: Extended Sound Library
## Table of Contents

## ProgAtr

Program header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Structure

**struct ProgAtr** {
    **unsigned char** *tones*;
    **unsigned char** *mvol*;
    **unsigned char** *prior*;
    **unsigned char** *mode*;
    **unsigned char** *mpan*;
    **char** *reserved0*;
    **short** *attr*;
    **unsigned long** *reserved1*;
    **unsigned long** *reserved2*;
};

### Members

| | |
|---|---|
| *tones* | Number of VAG attribute sets contained in the program |
| *mvol* | Master volume for the program |
| *prior* | Program priority (0-15) |
| *mode* | Sound source mode |
| *mpan* | Program pan |
| *reserved0* | Reserved by the system |
| *attr* | Program attribute |
| *reserved1* | Reserved by the system |
| *reserved2* | Reserved by the system |

### Explanation

### Remarks

**See also:**

## SndVolume

Volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Structure

**struct SndVolume** {
    **unsigned short** *left*;
    **unsigned short** *right*;
**};**

### Members

| | |
|---|---|
| *left* | L channel volume value |
| *right* | R channel volume value |

### Explanation

### Remarks

**See also:**

## VabHdr

Bank header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Structure

**struct VabHdr {**
  **long** *form*;
  **long** *ver*;
  **long** *id*;
  **unsigned long** *fsize*;
  **unsigned short** *reserved0*;
  **unsigned short** *ps*;
  **unsigned short** *ts*;
  **unsigned short** *vs*;
  **unsigned char** *mvol*;
  **unsigned char** *pan*;
  **unsigned char** *attr1*;
  **unsigned char** *attr2*;
  **unsigned long** *reserved1*;
**};**

### Members

| | |
|---|---|
| *form* | Format name (always 'VABp') |
| *ver* | Format version number |
| *id* | Bank (VAB) number |
| *fsize* | Bank file size |
| *reserved0* | Reserved by the system |
| *ps* | Total number of programs contained in the bank |
| *ts* | Total number of tones contained in the bank |
| *vs* | Number of VAGs contained in the bank |
| *mvol* | Master volume |
| *pan* | Master pan level |
| *attr1* | Bank attribute 1 that can be defined by the user |
| *attr2* | Bank attribute 2 that can be defined by the user |
| *reserved1* | Reserved by the system |

### Explanation

The VAB bank header contains information, such as sound source data set size and sound source numerals, that is used at the time of execution.

When SsVabOpenHead() is called, it is read by the system and wave form data is generated in the SPU's local memory. Also, volume setting and panning setting are referred at the time of voice allocation.

Information about VAB, the program and each VAG header can change at the time of execution by the user, and the attribute value is reflected in the voice application after the next KEY ON.

### Remarks

**See also:**

## VagAtr

Waveform header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Structure

**struct VagAtr** {
    **unsigned char** *prior*;
    **unsigned char** *mode*;
    **unsigned char** *vol*;
    **unsigned char** *pan*;
    **unsigned char** *center*;
    **unsigned char** *shift*;
    **unsigned char** *min*;
    **unsigned char** *max*;
    **unsigned char** *vibW*;
    **unsigned char** *vibT*;
    **unsigned char** *porW*;
    **unsigned char** *porT*;
    **unsigned char** *pbmin*;
    **unsigned char** *pbmax*;
    **unsigned char** *reserved1*;
    **unsigned char** *reserved2*;
    **unsigned short** *adsr1*;
    **unsigned short** *adsr2*;
    **short** *prog*;
    **short** *vag*;
    **short** *reserved* [*4*];
**};**

### Members

| | |
|---|---|
| *prior* | Priority (0-127) |
| *mode* | Sound source mode (Bit values 0: normal, 1: reverb) |
| *vol* | Volume (0-127, 0:min, 127:max) |
| *pan* | Pan pot (0-127, 0:left, 63:center, 127:right) |
| *center* | Center note (0-127) |
| *shift* | Pitch correction (0-99, in cents) (center note fine tune) |
| *min* | Minimum note limit |
| *max* | Maximum note limit |
| *vibW* | Vibrato width (0-127 over one octave) |
| *vibT* | Period of vibrato cycle (in ticks) |
| *porW* | Portamento width |
| *porT* | Period of portamento duration (in ticks) |
| *pbmin* | Minimum pitch bend limit |
| *pbmax* | Maximum pitch bend limit |
| *reserved1* | Reserved by the system |
| *reserved2* | Reserved by the system |
| *adsr1* | Set ADSR value 1 |
| *adsr2* | Set ADSR value 2 |
| *prog* | Master program containing the VAG attribute |
| *vag* | VAG's ID number utilized by the VAG attribute |
| *reserved* [*0...3*] | Reserved by the system |

## Explanation

## Remarks

**See also:**

## _SsFCALL*

Function table type referenced in SsSeqOpenJ() and SsSepOpenJ().

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.6* | *10/23/96* |

### Structure

```
typedef struct {
    void (*noteon) ();
    void (*programchange) ();
    void (*pitchbend) ();
    void (*metaevent) ();
    void (*control[13]) ();
    void (*ccentry[20]) ();
} _SsFCALL;
```

### Members

All members hold pointers to the low level functions.

| | |
|---|---|
| *noteon, programchange, pitchbend, metaevent, control, ccentry* | MIDI status data |
| (control array | Events of MIDI status data) |
| (control change, ccentry array | Entry events for nrpn, rpn data) |

### Explanation

Functions SsSeqPlay() and SsSepPlay() analyze the MIDI status data and call low level functions. Although there are many low level functions, an application would not usually use all the functions. These low level function groups will be set by calling either SsSeqOpen() or SsSepOpen(). In order to reduce the code size by not linking unnecessary low level functions, new functions SsSeqOpenJ() and SsSepOpenJ(), compatible with SsSeqOpen() and SsSepOpen() respectively. In the new functions, low level functions are in the jump table so that the user can set only desired function group.

_SsFCALL is a structure that defines this function table.Necessary function can be linked by assigning the pointer to the low level function. In reverse, link can be eliminated by not assigning pointer and not placing extern declaration. Note that calling a function without setting a pointer will cause BUS ERROR. To avoid BUS ERROR, set a dummy function by prefixing the low level function name with "dmy".

### Return value

### Remarks

**See also:** SsVoKeyOff (p. 12-114).

## dmy_Ss*

Jump table low level function dummy.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.6* | *10/23/96* |

### Syntax

**void dmy_Ss***...()*

**Arguments**

None.

**Explanation**

When this function is called for the first time, it outputs the entry name of the jump table to the standard output device. Use this as a dummy low level function and to determine which entry was called.

**Return value**

None.

**Remarks**

This function is provided for debugging.

**See also:**

## SsChannelMute*

Select SEQ channel and play.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.6* | *10/23/96* |

### Syntax

**void SsChannelMute** *(***short** *acn,* **short** *trn,* **long** *channels)*

### Arguments

| | |
|---|---|
| acn | SEP access number |
| *trn* | SEQ number within SEP<br>(0 when the music score data is SEQ |
| *channels* | MIDI channel |

### Explanation

This function specifies MIDI channel in SEQ with 16bit upon playing SEQ. The parts specified with the channel bits can be muted. This function must be called before SsSeqOpen() or SsSepOpen().

### Return value

None.

### Remarks

**See also:** SsSeqPlay(), SsSepPlay().

## SsEnd

Stops the sound system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsEnd** (void)

### Arguments

None.

### Explanation

If SsSetTickMode() is used to set the mode that automatically calls SsSeqCalledTbyT(), this function, after it is called, stops SsSeqCalledTbyT() from being called for every Tick.

### Return value

None.

### Remarks

**See also:** SsStart, SsSetTickMode (p. 12-62), SsSeqCalledTbyT (p. 12-31), SsQuit (p. 12-23).

## SsGetCurrentPoint*

Obtain SEQ/SEP address currently read-in.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.6* | *10/23/96* |

### Syntax

**unsigned char* SsGetCurrentPoint** *(***short** *acn,* **short** *trn)*

### Arguments

| | |
|---|---|
| acn | SEP access number |
| *trn* | SEQ number within SEP<br>(0 when the music score data is SEQ |

### Explanation

This function obtains the current read-in address for the SEQ/SEP data that is being played.

### Return value

SEP/SEQ data address.

### Remarks

**See also:** SsSeqPlay(), SsSepPlay().

## SsGetMute

Obtains mute attribute.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**char SsGetMute** *(***void***)*

### Arguments

None*.*

### Explanation

This function obtains the mute attribute.

### Return value

SS_MUTE_ON ... Mute on

SS_MUTE_OFF ... Mute off

### Remarks

**See also:** SsSetMute (p. 12-55).

## SsGetMVol

Main volume value acquisition.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsGetMVol** (*m_vol*)
**SndVolume** **m_vol*;

### Arguments

| | |
|---|---|
| *m_vol* | Pointer to main volume value |

### Explanation

Returns the main volume value to *m_vol*.

### Return value

None.

### Remarks

**See also:** SsSetMVol (p. 12-46).

## SsGetNck

Gets noise clock value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.1* | *7/31/96* |

### Syntax

**short SsGetNck** (void)

### Arguments

None.

### Explanation

Returns the noise clock value.

### Return value

Noise clock value.

### Remarks

**See also:** SsSetNck (p.).

## SsGetRVol

Gets reverb volume value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**Void SsGetRVol** (*r_vol*)
**SndVolume** **r_vol*;

### Arguments

| | |
|---|---|
| *r_vol* | Pointer to reverb volume value |

### Explanation

Returns the reverb volume value to *r_vol*.

### Return value

None.

### Remarks

**See also:** SsSetRVol (p. 12-62).

## SsGetSerialAttr

Gets a serial attribute value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**char SsGetSerialAttr** (*s_num*, *attr*)
**char** *s_num*;
**char** *attr*;

### Arguments

| | |
|---|---|
| s_num | Serial Number |
| attr | Attribute |

### Explanation

Returns the specified serial attribute value.

(a) *s_num*

**Table 12–1**

| Macro | Contents |
|---|---|
| SS_SERIAL_A | Serial A (CD input) |
| SS_SERIAL_B | Serial B (external digital input) |

(b) *attr*

**Table 12–2**

| Macro | Contents |
|---|---|
| SS_MIX | Mixing |
| SS_REV | Reverb |

### Return value

Attribute: Returns 1 if on. Returns 0 if off.

### Remarks

**See also:** SsSetSerialAttr (p. 12-56).

## SsGetSerialVol

Gets a serial volume value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.1* | *7/31/96* |

### Syntax

**void SsGetSerialVol** (*s_num*, s_*vol*)
**char** *s_num*;
**SndVolume** **s_vol*;

### Arguments

| | |
|---|---|
| *s_num* | Serial number |
| *s_vol* | Pointer to volume value |

### Explanation

Returns the specified serial number volume value.

**Table 12–3**

| Macro | Contents |
|---|---|
| SS_SERIAL_A | Serial A (CD input) |
| SS_SERIAL_B | Serial B (external digital input) |

### Return value

None.

### Remarks

**See also:** SsSetSerialVol (p.).

## SsInit

Initializes sound system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsInit** (void)

### Arguments

None.

### Explanation

This function initializes the sound system, clearing the sound local memory.

### Return value

None.

### Remarks

**See also:** SsInitHot (p.), SsEnd (p. 12-6), SpuInit (see libspu).

## SsInitHot

Initializes sound system (hot reset).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.1* | *7/31/96* |

### Syntax

**void SsInitHot** (void)

### Arguments

None.

### Explanation

This function performs initialization of the sound system without destroying the data that has been transferred to the sound buffer. Using Exec-related functions, when you want to initialize the sound system by a child process, with the sound buffer in its current state, the child process should call SslnitHot instead of calling SsInit as it normally would.

### Return value

None.

### Remarks

**See also:** SsInit (), Exec-related functions, SpuInitHot (
.).

## SsIsEos

Determines whether or not a song is being played.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsIsEos** (*access_num*, *seq_num*)
**short** *access_num*;
**short** *seq_num*;

### Arguments

| | |
|---|---|
| *access_num* | SEQ/SEP access number |
| *seq_num* | SEQ number inside SEP data |

### Explanation

Determines whether or not a specified song is being played.

When using this function for SEQ data, set 0 in *seq_num*; when using this function for SEP data, set the number that contains the SEQ to be played.

### Return value

Returns 1 if the song is being played.

Returns 0 if the song is not being played.

### Remarks

**See also:**

## SsPlayBack

Reads SEQ/SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsPlayBack** (*access_num*, *seq_num*, *l_count*)
**short** *access_num*;
**short** *seq_num*;
**short** *l_count*;

### Arguments

| | |
|---|---|
| *access_num* | SEQ/SEP access number |
| *seq_num* | SEQ number inside SEP data |
| *l_count* | Song repetition count |

### Explanation

In the current play mode, no event occurs when a function is called again during execution. However, this function, if called again during execution, stops the song being played, returns to the start of the song, and begins playing it again.

When using this function for SEQ data, set 0 in seq_num; when using this function for SEP data, set the number that contains the SEQ to be played. Specify a song repetition count in l_count.

For infinite play repetition, specify SSPLAY_INFINITY.

### Return value

None.

### Remarks

**See also:** SsSeqPlay (p. 12-36), SsSepPlay (p. 12-23).

## SsQuit

Terminate the sound system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsQuit** (*void*)

### Arguments

None.

### Explanation

Terminates the sound system. After this function is called, transfer to the sound buffer will be disabled. To enable transfer to the sound buffer again, SsInit () must be called.

SsEnd () must be called before SsQuit ().

### Return value

None.

### Remarks

**See also:** SsEnd (p. 12-1), SsStart (p. 12-64), SsSetTickMode (p. 12-62), SsSeqCalledTbyT (p. 12-31).

## SsSepClose

Close SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepClose** (*sep_access_num*)
**short** *sep_access_num*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |

### Explanation

Closes SEP data possessing *sep_access_num* that is no longer needed.

Because closing is performed on a SEP unit basis, all SEQ data stored in the closed SEP will become inaccessible.

### Return value

### Remarks

**See also:** SsSepOpen (p. 12-20).

## SsSepOpen

OpenOpens SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsSepOpen** (*addr*, *vab_id*, *seq_num*)
**unsigned long** **addr*;
**short** *vab_id*;
**short** *seq_num*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to starting address of SEP data within the main memory |
| *vab_id* | VAB id |
| *seq_num* | Number of SEQs contained in SEP |

### Explanation

Analyzes the SEP data located in the main memory, and returns a SEP access number. Maximum of 32 pieces of SEP data can be opened simultaneously when combined with the number of open SEQ data.

### Return value

SEP access number.

An internal SEP data management table number that possesses the same characteristics as the SEQ access number.

### Remarks

**See also:** SsSepClose (p. 12-24).

## SsSepOpenJ*

Opens SEP data (function table version).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.6* | *10/23/96* |

### Syntax

**short SsSepOpenJ** *(***unsigned long** **addr*; **short** *vab_id*; **short** *num2;)*

### Arguments

| | |
|---|---|
| *addr* | Pointer to starting address of SEP data within the main memory |
| *vab_id* | VAB id |
| *num2* | Number of SEQs contained in SEP |

### Explanation

This function is equivalent to SsSepOpen() if all the low level functions were registered. In addition to the SsSepOpen() capability, this function enables a programmer to control functions to be registered to the table and thus improve code efficiency by not calling unnecessary low level functions.

For those slots that SsFCALL will not register, use dummy functions, standard function names with the prefix "dmy" so that even when a lower function was called, no BUS ERROR would occur and the function names would be printed out. After checking the called function names, register the function names without "dmy".

### Return value

SEQ Access Number: Used in the SEQ data access function, being the inner SEQ data control table number.

### Remarks

**See also:** SsFCALL function table.

## SsSepPause

Pause the reading of SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepPause** (*sep_access_num*, *seq_num*)
**short** *sep_access_num*;
**short** *seq_num*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |
| *seq_num* | SEQ number inside SEP data |

### Explanation

Pauses the reading (playing) of the *seq_num* SEQ data of SEP data possessing *sep_access_num*.

### Return value

None.

### Remarks

**See also:** SsSepReplay (p. 12-23).

## SsSepPlay

Reads (plays) SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepPlay** (*sep_access_num*, *seq_num*, *play_mode*, *l_count*)
**short** *sep_access_num*;
**short** *seq_num*;
**char** *play_mode*;
**short** *l_count*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |
| *seq_num* | SEP data SEQ number |
| *play_mode* | Play mode |
| *l_count* | Song repetition count |

### Explanation

Begins to read (play) SEQ data specified by the SEP data *seq_num* specified by *seq_access_num*, or, depending on the *play_mode* value, you may choose a pause state. For infinite play repetition, specify SSPLAY_INFINITY.

**Table 12–4**

| Play_mode | Actions |
|---|---|
| SSPLAY_PAUSE | Makes a pause state |
| SSPLAY_PLAY | Plays immediately |

Examples:

(1) Opens SEP data containing four pieces of SEQ data:
sep1 = SsSepOpen (addr, vab_id, 4);
(2) Immediately plays the third piece of data of the opened SEP data twice.
SsSepPlay (sep1, 2, SSPLAY_PLAY, 2);

### Return value

None.

### Remarks

**See also:** SsSepStop (p. 12-35).

## SsSepReplay

Resume (replay) the reading of SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepReplay** (*sep_access_num*, *seq_num*)
**short** *sep_access_num*;
**short** *seq_num*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |
| *seq_num* | SEQ number inside SEP data |

### Explanation

Resumes the reading of the *seq_num* SEQ data of SEP data possessing *sep_access_num*, that was paused by SsSepPause.

### Return value

None.

### Remarks

**See also:** SsSepPause ().

## SsSepSetAccelerando

Accelerate the tempo.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepSetAccelerando** (*sep_access_num*, *seq_num*, *tempo*, *v_time*)
**short** *sep_access_num*;
**short** *seq_num*;
**long** *tempo*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEQ access number |
| *seq_num* | SEQ number inside SEP data |
| *tempo* | Song tempo |
| *v_time* | Time (in ticks) |

### Explanation

Increases the tempo of the *seq_num*-th SEQ data of SEP data possessing *sep_access_num* down to tempo within *v_time*.

However, if the specified tempo is smaller (slower) than the current tempo, this function acts the same as SsSepSetRitardando.

### Return value

None.

### Remarks

**See also:** SsSepSetRitardando (p. 12-33).

## SsSepSetCrescendo

Crescendo (valid for individual SEQ in SEP).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepSetCrescendo** (*sep_access_num*, *seq_num*, *vol*, *v_time*)
**short** *sep_access_num*;
**short** *seq_num*;
**short** *vol*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |
| *seq_num* | SEQ number inside SEP data |
| *vol* | Volume value (0-127) |
| *v_time* | Time (in tick units) |

### Explanation

Raises the main volume of the *seq_num* SEQ data of SEP data possessing *sep_access_num* by *vol* within *v_time*.

Note that this function will have no effect if the volume of each voice is at the maximum or if *vol* is a negative number.

### Return value

None.

### Remarks

**See also:** SsSepSetDecrescendo ().

## SsSepSetDecrescendo

Decrescendo (valid for individual SEQ in SEP).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepSetDecrescendo** (*sep_access_num*, *seq_num*, *hort vol*, *v_time*)
**short** *sep_access_num*;
**short** *seq_num*;
**short** *vol*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |
| *seq_num SEQ* | Number inside SEP data |
| *vol* | Volume value (0-127) |
| *v_time* | Time (in tick units) |

### Explanation

Lowers the main volume of the *seq_num* SEQ data of SEP data possessing *sep_access_num* by *vol* within *v_time*.

Note that this function will have no effect if the volume of each voice is at the minimum or if *vol* is a negative number.

### Return value

None.

### Remarks

**See also:** SsSepSetCrescendo ().

## SsSepSetRitardando

Slows the tempo.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepSetRitardando** (*sep_access_num*, *seq_num*, *tempo*, *v_time*)
**short** *sep_access_num*;
**short** *seq_num*;
**long** *tempo*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEQ access number |
| *seq_num* | SEQ number inside SEP data |
| *tempo* | Song tempo |
| *v_time* | Time (in tick units) |

### Explanation

Slows the tempo of the *seq_num* SEQ data of SEP data possessing *sep_access_num* down to tempo within *v_time*.

However, if the specified tempo is larger (faster) than the current tempo, this function acts the same as SsSepSetAccelerando.

### Return value

None.

### Remarks

**See also:** SsSepSetAccelerando (p. 12-25).

## SsSepSetVol

SEP volume setting (valid for individual SEQ in SEP).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepSetVol** (*sep_access_num*, *seq_num*, *voll*, *volr*)
**short** *sep_access_num*;
**short** *seq_num*;
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |
| *seq_num* | SEQ number inside SEP data |
| *voll* | L channel main volume value |
| *volr* | R channel main volume value |

### Explanation

Sets the L and R channels for the main volume of the *seq_num* SEQ data of SEP data possessing *sep_access_num* to specified values.

A value between 0 and 127 can be set.

### Return value

None.

### Remarks

**See also:**

## SsSepStop

Stops the reading of SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSepStop** (*sep_access_num*, *seq_num*)
**short** *sep_access_num*;
**short** *seq_num*;

### Arguments

| | |
|---|---|
| *sep_access_num* | SEP access number |
| *seq_num* | SEQ number inside SEP data |

### Explanation

Terminates the reading (playing) of the *seq_num* SEQ data of SEP data possessing *sep_access_num*.

### Return value

None.

### Remarks

**See also:** SsSepPlay (p. 12-23).

## SsSeqCalledTbyT

It is called at each 1 Tick, interprets SEQ/SEP data and carries out playingback.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqCalledTbyT** (void)

### Arguments

None.

### Explanation

At each Tick this function is called; it interprets SEQ/SEP data and carries out playback. Tick is set by SsSetTickMode(), but this Tick merely regulates the internal sound system, without depending either on the speed or resolution determined by SEQ/SEP data.

When SsSetTickMode is specified, the sound system calls this function with the given resolution if the tick_mode is macro SS_TICK60 or SS_TICK240. However, if SS_NOTICK is specified, this function must be called by the program at each 1/60 second interval (usually with vertical sync (VSync()) timing).

### Return value

None.

### Remarks

**See also:** SsSetTickMode (p. 12-62).

## SsSeqClose

Closes SEQ data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqClose** (*seq_access_num*)
**short** *seq_access_num*;

### Arguments

*seq_access_num* SEQ access number

### Explanation

This function closes SEQ data with an un-needed *seq_access_num*.

### Return value

None.

### Remarks

**See also:** SsSeqOpen ().

## SsSeqGetVol

Obtaining SEQ volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqGetVol** (**short** *access_num*, **short** *seq_num*, **short** **voll*, **short** **volr*)

### Argument

| | |
|---|---|
| *access_num* | SEQ/SEP access number |
| *seq_num* | SEQ number of SEP data |
| *voll* | L volume of SEQ data |
| *volr* | R volume of SEQ data |

### Explanation

This function returns current left and right SEQ volume to voll and volr. Set seq_num at 0 for SEQ data, and set it at appropriate SEQ number for SEP data.

The volume value set by SsSepSetVol() can be obtained by this function.

### Return value

None.

### Remarks

**See also:** SsSeqSetVol (p. 12-49), SsSepSetVol (p. 12-34).

## SsSeqOpen

Opens SEQ data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsSeqOpen** (*addr*, *vab_id*)
**unsigned long** **addr*;
**short** *vab_id*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to start address of SEQ data in the main storage |
| *vab_id* | VAB id |

### Explanation

This function analyzes SEQ data in the main memory to return the SEQ access number.

### Return value

SEQ access number: because this is used in the SEQ data access function, this is the SEQ data control table number.

When you try to open more than 32 SEP data (combined with SEQ data) at the same time, the return value will be -1.

### Remarks

**See also:** SsSeqClose ().

## SsSeqOpenJ*

Opens SEQ data (function table version).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.6* | *10/23/96* |

### Syntax

**short SsSeqOpenJ** *(***unsigned long** **addr*; **short** *vab_id*; **short** *num2;)*

### Arguments

| | |
|---|---|
| *addr* | Pointer to start address of SEQ data in the main storage |
| *vab_id* | VAB id |

### Explanation

This function is equivalent to SsSeqOpen() if all the low level functions were registered. In addition to the SsSeqOpen() capability, this function enables a programmer to control functions to be registered to the table and thus improve code efficiency by not calling unnecessary low level functions.

For those slots that SsFCALL will not register, use dummy functions, standard function names with the prefix "dmy" so that even when a lower function was called, no BUS ERROR would occur and the function names would be printed out. After checking the called function names, register the function names without "dmy".

### Return value

SEQ Access Number: Used in the SEQ data access function, being the inner SEQ data control table number.

### Remarks

**See also:** SsFCALL function table.

## SsSeqPause

Pauses SEQ data reading.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqPause** (*seq_access_num*)
**short** *seq_access_num*;

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |

### Explanation

This function stops reading (playing) SEQ data with *seq_access_num*.

### Return value

None.

### Remarks

**See also:** SsSeqReplay ().

## SsSeqPlay

Reads (plays) SEQ data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqPlay** (*seq_access_num*, *play_mode*, *l_count*)
**short** *seq_access_num*;
**char** *play_mode*;
**short** *l_count*

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |
| *play_mode* | Performance mode |
| *l_count* | Number of repeats of the music |

### Explanation

This function selects either immediate SEQ data reading or sets a pause state at the start of SEQ data. Designate repeat play of the music by *l_count*, using SSPLAY_INFINITY if play is unlimited. For play mode, the parameters below may be specified.

**Table 12–5**

| Macro | State |
|---|---|
| SSPLAY PAUSE | Pause at start of piece |
| SSPLAY PLAY | Immediate performance |

### Return value

None.

### Remarks

**See also:** SsSeqPause (p. 12-41).

## SsSeqReplay

Resumes SEQ data reading (Replay) .

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqReplay** (*seq_access_num*)
**short** *seq_access_num*;

### Arguments

*seq_access_num* SEQ access number

### Explanation

This function resumes reading SEQ data with *seq_access_num* stopped by SsPause

### Return value

None.

### Remarks

**See also:** SsSeqPause ().

## SsSeqSetAccelerando

Quickens tempo.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqSetAccelerando** (*seq_access_num*, *tempo*, *v_time*)
**short** *seq_access_num*;
**long** *tempo*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |
| *tempo* | Music tempo |
| *v_time* | Time (in ticks) |

### Explanation

This function quickens the SEQ data with *seq_access_num* to the tempo resolution in *v_time*. With the specified resolution smaller (slower) than the current resolution, the function provides the same effect as SsSeqSetRitardando.

### Return value

None.

### Remarks

**See also:** SsSeqSetRitardando ().

## SsSeqSetCrescendo

Crescendo.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqSetCrescendo** (*seq_access_num*, *vol*, *v_time*)
**short** *seq_access_num*;
**short** *vol*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |
| *vol* | Volume value (0-127) |
| *v_time* | Time (in ticks) |

### Explanation

This function increases the main volume of SEQ data with *seq_access_num* by the vol value in *v_time*. With the maximum voice volume, or if *vol* is a negative number, the function provides no effect.

### Return value

None.

### Remarks

**See also:** SsSeqSetDecrescendo (p. 12-40).

## SsSeqSetDecrescendo

Decrescendo.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqSetDecrescendo** (*seq_access_num*, *vol*, *v_time*)
**short** *seq_access_num*;
**short** *vol*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |
| *vol* | Volume value (0-127) |
| *v_time* | Time (in ticks) |

### Explanation

Lowers main volume of SEQ data with *seq_access_num* by the *vol* valve in *v_time*. If each voice volume is the maximum value, or if *vol* is a negative number, there is no effect.

### Return value

None.

### Remarks

**See also:** SsSeqSetCrescendo (p. 12-45).

## SsSeqSetNext

Specifies subsequent SEQ data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqSetNext** (*seq_access_num1*, *seq_access_num2*)
**short** *seq_access_num1*;
**short** *seq_access_num2*;

### Arguments

| | |
|---|---|
| *seq_access_num1* | SEQ access number |
| *seq_access_num2* | SEQ access number |

### Explanation

This function specifies the SEQ access number (*seq_access_num2*) of SEQ data to be performed after the SEQ data with *seq_access_num1*.

### Return value

None.

### Remarks

**See also:** SsSetNext (p. 12-58)

## SsSeqSetRitardando

Slows tempo.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqSetRitardando** (*seq_access_num*, *tempo*, *v_time*)
**short** *seq_access_num*;
**long** *tempo*;
**long** *v_time*;

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |
| *tempo* | Music tempo |
| *v_time* | Time (in ticks) |

### Explanation

This function slows the SEQ data with *seq_access_num* to the tempo resolution *in v_time*. With the specified resolution larger (faster) than the current resolution, however, the function provides the same effect as SsSeqSetAccelerando.

### Return value

None.

### Remarks

**See also:** SsSeqSetAccelerando (p. 12-38).

## SsSeqSetVol

Sets SEQ volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSeqSetVol** (*seq_access_num*, *voll*, *volr*)
**short** *seq_access_num*;
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |
| *voll L* | Channel’s main volume value |
| *volr R* | Channel’s main volume value |

### Explanation

This function sets the main volume of music with *seq_access_num* at values specified for the L and R channels. *voll* and *volr* range from 0 to 127.

### Return value

None.

### Remarks

**See also:**

## SsSeqStop

Terminates SEQ data reading.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSeqStop** (*seq_access_num*)
**short** *seq_access_num*;

### Arguments

| | |
|---|---|
| *seq_access_num* | SEQ access number |

### Explanation

This function terminates the reading of SEQ data with *seq_access_num* (performance).

### Return value

None.

### Remarks

**See also:** SsSeqPlay (p. 12-36).

## SsSetAutoKeyOffMode

Sets the automatic KeyOff mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSetAutoKeyOffMode** (*mode*)
**short** *mode*;

### Arguments

| | |
|---|---|
| *mode 0* | Automatically keys off. |
| *mode 1* | Does not key off until a KeyOff request comes in. |

### Explanation

Sets the automatic KeyOff mode. The default is the automatic KeyOff mode. If the envelopes for the past 16 interrupts contain all 0's, the automatic KeyOff mode assumes that waveform playback has been automatically terminated, and uses the voice for other waveform playback.

### Return value

None.

### Remarks

**See also:**

## SsSetLoop

Sets a song repetition count.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSetLoop** (*access_num*, *seq_num*, *l_count*)
**short** *access_num*;
**short** *seq_num*;
**short** *l_count*;

### Arguments

| | |
|---|---|
| *access_num* | SEQ/SEP access number |
| *seq_num* | SEQ number inside SEP data |
| *l_count* | Song repetition count |

### Explanation

Sets a song repetition count.This function is useful for changing the song repetition count set in SsSeqPlay. After this function is called, the current song repetition count will be reset, and the song will be played for the number of times set by the new count.

When using this function for SEQ data, set 0 in *seq_num*; when using this function for SEP data, set the number that contains the SEQ to be played.

### Return value

None.

### Remarks

**See also:**

## SsSetMarkCallback

Register a function to be called when a mark is detected.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**typedef void (*SsSeqMarkCallbackProc)** (*short*, *short*, *short*)
**void SsSetMarkCallback** (*access_num*, *seq_num*, *proc*)
**short** *access_num*;
**short** *seq_num*;
**SsMarkCallbackProc** *proc*;

### Arguments

| | |
|---|---|
| *access_num* | SEQ/SEP access number |
| *seq_num* | SEQ number inside SEP data |
| *proc* | Callback function to be called when Mark is detected |

### Explanation

When a mark is detected inside a song possessing *access_num*, a Callback function will be called. During this operation, SEQ/SEP number will be handed over to the first argument; SEQ number inside SEP data will be handed over to the second argument; and the data2 value set in Mark will be handed over to the third argument. Set the second argument to 0 when using SEQ. The function clears the Callback function when NULL is given to proc.

Only one Callback function can be registered at a time.

Sample

```
/* Callback function-definition*/
SsMarkCallbackProc proc (short ac_no, short tr_no, short
data);
:
/* Opens SEQ */
short seq_a_num = SsSeqOpen (addr, vab_id);
/* Sets Callback function */
SsSetMarkCallback (seq_a_num, 0, (SsMarkCallbackProc) proc);
```

### Return value

None.

### Remarks

**See also:**

## SsSetMono

Set monaural mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetMono** (*void*)

### Arguments

None.

### Explanation

Sets the output to monaural mode.

### Return value

None.

### Remarks

**See also:** SsSetStereo ()

## SsSetMute

Set a Mute.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetMute** (*mode*)
**char** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Setting mode |

### Explanation

This function sets a mute. The values below may be specified for *mode*.

**Table 12–6**

| Macro | Contents |
|---|---|
| SS_MUTE_ON | Mute on |
| SS_MUTE_OFF | Mute off |

### Return value

None.

### Remarks

**See also:**

## SsSetMVol

Set main volume value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetMVol** (*voll*, *volr*)
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *voll* | L channel volume value |
| *volr* | R channel volume value |

### Explanation

This function sets the main volume values for *voll* and *volr*. The value ranges from 0 to 127.

You must set this before playing sequence (SEQ, SEP) data.

### Return value

None.

### Remarks

**See also:** SsGetMVol (p. 12-14).

## SsSetNck

Sets noise clock value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.1* | *7/31/96* |

### Syntax

**void SsSetNck** (*n_clock*)
**short** *n_clock*;

### Arguments

| | |
|---|---|
| *n_clock* | Noise clock value |

### Explanation

Sets the noise clock value. The possible values are from 0 - 0x3c. Noise is lower if the value is smaller, and louder if the value is larger.

### Return value

None.

### Remarks

**See also:** SsGetNck (p. 12-10).

## SsSetNext

This function sets the next SEQ/SEP data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.2* | *7/31/96* |

### Syntax

**void SsSetNext** (**short** *ac_no1*, **short** *tr_no1*, **short** *ac_no2*, **short** *tr_no2*)

### Arguments

| | |
|---|---|
| *ac_no1* | SEP/SEQ access number |
| *tr_no1* | SEQ number in SEP (If the score data is SEQ, tr_no1 is 0.) |
| *ac_no2* | SEP/SEQ access number |
| *tr_no2* | SEQ number in SEP (If the score data is SEQ, tr_no2 is 0.) |

### Explanation

This function sets the score data with SEP/SEQ access numbers (ac_no2, tr_no2) to be played after SEP/SEQ data (ac_no1, tr_no1).

The next score data is played automatically after the previous score finishes playing.

### Return value

None.

### Remarks

**See also:** SsSeqSetNext (p. 12-47).

## SsSetNoiseOff

Sets Noise off.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetNoiseOff** (*void*)

### Arguments

None.

### Explanation

Makes Noise Off

### Return value

None.

### Remarks

**See also:** SsSetNoiseOn ().

## SsSetNoiseOn

Sets Noise on.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetNoiseOn** (*voll*, *volr*)
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *voll* | L channel volume value |
| *volr* | R channel volume value |

### Explanation

Makes Noise On with the given volume value. Volume values may be between 0-127. It sets the Noise Clock value with SsSetNck before making Noise On.

### Return value

None.

### Remarks

**See also:** SsSetNoiseOff (p. 12-59).

## SsSetReservedVoice

Declares the number of voices to be allocated by libsnd library.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**char SsSetReservedVoice** (*voices*)
**char** *voices*;

### Arguments

| | |
|---|---|
| *voices* | Voice count |

### Explanation

Declares the number of voices libsnd library will use for allocation. Other voices can be used in libspu by the user. (They must always be called in "all key off.")

For example, if char = 20, then:

- Voices 0-19 will be used for allocation by libsnd.
- Voices 20-23 will be available for libspu.

### Return value

Returns the set voice count if successful. Returns -1 if unsuccessful.

### Remarks

**See also:**

## SsSetRVol

Sets reverberant volume values.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetRVol** (*voll*, *volr*)
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *voll* | L channel's volume value |
| *volr* | R channel's volume value |

### Explanation

This function sets the reverberant volume values for *voll* and *volr*. The value ranges from 0 to 127.

### Return value

None.

### Remarks

**See also:** SsGetRVol (p. 12-16).

## SsSetSerialAttr

Sets a serial attribute.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### *ynt***ax**

**void SsSetSerialAttr** (*s_num*, *attr*, *mode*)
**char** *s_num*;
**char** *attr*;
**char** *mode*;

### Arguments

| | |
|---|---|
| *s_num* | Serial number |
| *attr* | Attribute value |
| *mode* | Setting mode |

### Explanation

Sets a serial attribute.

(a) *s_num*

**Table 12–7**

| Macro | Contents |
|---|---|
| SS_SERIAL_A | Serial A (CD input) |
| SS_SERIAL_B | Serial input line B (external digital input) |

(b) *attr*

**Table 12–8**

| Macro | Contents |
|---|---|
| SS_MIX | Mixing |
| SS_REV | Reverb |

(c) *mode*

**Table 12–9**

| Macro | Contents |
|---|---|
| SS_SON | attr on |
| SS_SOFF | attr off |

### Return value

None.

### Remarks

**See also:** SsGetSerialAttr (p. 12-17).

## SsSetSerialVol

Sets a serial volume value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetSerialVol** (*s_num*, *voll*, *volr*)
**char** *s_num*;
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *s_num* | Serial number |
| *voll* | L channel volume value |
| *volr* | R channel volume value |

### Explanation

Sets the value of the serial volume in *voll*, *volr*. The volume values may be set between 0-127.

**Table 12–10**

| Macro | Contents |
|---|---|
| SS_SERIAL_A | Serial A (CD input) |
| SS_SERIAL_B | Serial B (external digital input) |

### Return value

None.

### Remarks

**See also:** SsGetSerialVol ().

## SsSetStereo

Sets stereo mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetStereo** (*void*)

### Arguments

None.

### Explanation

Sets the output to stereo mode. The sound system default output is stereo.

### Return value

None.

### Remarks

**See also:** SsSetMono ().

## SsSetTableSize

Specifies the area of a SEQ/SEP data attribute table.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSetTableSize** (*table*, *s_max*, *t_max*)
**char** **table*;
**short** *s_max*;
**short** *t_max*;

### Arguments

| | |
|---|---|
| *table* | Pointer to SEQ/SEP data attribute table area variable |
| *s_max* | Maximum frequency of opening SEQ/SEP data |
| *t_max* | Number of SEQ included in SEP |

### Explanation

The area of a SEQ/SEP data attribute table is set in the library. The library uses this area to analyze SEQ/SEP data, then saves it and plays it back.

*s_max* specifies the maximum number of times SEQ/SEP data may be opened. The upper limit is 32. Once the upper limit is reached, unused SEQ/SEP data must be closed with SsSeqClose/SsSepClose before more data can be opened. *t_max* specifies the number of SEQ included in the SEP data. Set *t_max* to 1 to handle only SEQ data and not use SEP data. The upper limit of t_max is 16.

In table, you must preserve the area by using global variables or functions like malloc() (auto variables cannot be used in a function).

Use the following to find the size from the library:
  (SS_SEQ_TABSIZ x s_max x t_max)

where the constant SS_SEQ_TABSIZ is declared in libsnd.h. Note that the value of this constant varies from version to version, so use the constant when saving the table area.

SsSetTableSize() is called immediately after SsInit(). Both functions are set to be called only once; what happens when multiple calls are made is unclear.

### Return value

None.

### Remarks

**See also:** SsSeqClose (p. 12-37), SsSepClose (p. 12-24), SsSeqOpen (p. 12-39), SsSepOpen (p. 12-25).

## SsSetTempo

Set a tempo.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsSetTempo** (*access_num*, *seq_num*, *tempo*)
**short** *access_num*;
**short** *seq_num*;
**short** *tempo*;

### Arguments

| | |
|---|---|
| *access_num* | SEQ/SEP access number |
| *seq_num* | SEQ number inside SEP data |
| *tempo* | Song tempo |

### Explanation

Sets a tempo. This function is useful for changing the tempo set in SsSeqPlay.

After this function is called, the current tempo will be changed to the new tempo specified for playing songs.

When using this function for SEQ data, set 0 in *seq_num*; when using this function for SEP data, set the number that contains the SEQ to be played.

### Return value

None.

### Remarks

**See also:**

## SsSetTickCallBack

Sets the TickCallBack function called with every TICK.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**int SsSetTickCallback (**
**void** (**cb*)()

### Arguments

| | |
|---|---|
| *cb* | Pointer to TickCallBack function called with every Tick |

### Explanation

Sets the TickCallBack function called with every Tick. Only when SS_NOTICK has not been set on SsSetTickMode, after SsStart () or SsStart2 () have been called, function *cb* will be called with each Tick.

When tick Callback function has not been set using SsSetTickCallBack, the default will be set as SsSeqCalledTbyT ().

### Return value

Previously-set TickCallback function.

### Remarks

**See also:** SsSetTickMode (p. 12-69), SsStart (p. 12-71), SsStart2 (p. 12-72), SsSeqCalledTbyT (p. 12-36).

## SsSetTickMode

Sets Tick.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsSetTickMode** (*tick_mode*)
**long** *tick_mode*;

### Arguments

*tick_mode* Tick Mode

### Explanation

Sets the resolution of Tick. Call this function only once before calling SsSeqOpen() or SsSepOpen() for the first time. When it is called multiple times, correct operation cannot be guaranteed.

Tick Mode does not depend on the speed or resolution specified by SEQ/SEP data, and merely specifies the resolution inside the sound system.

In Tick Mode, the effects of SS_TICK50, SS_TICK60, and SS_TICKVSYNC differ according to the specification of SetVideoMode() (see the individual Tick Mode entries below).

*tick_mode* may be specified with the following values.

**Table 12–11**

| tick_mode | Setting |
|---|---|
| SS_TICK50<br>SsSeqCalledTbyT | Tick = 1/50 second<br>Automatic call |
| SS_TICK60<br>SsSeqCalledTbyT | Tick = 1/60 second<br>Automatic call |
| SS_TICKVSYNC<br>SsSeqCalledTbyT | Tick = resolution of vertical sync<br>Automatic call |
| SS_TICK120<br>SsSeqCalledTbyT | Tick = 1/120 second<br>Automatic call |
| SS_TICK240<br>SsSeqCalledTbyT | Tick = 1/240 second<br>Automatic call |
| SS_NOTICK<br>SsSeqCallTbyT | Tick = 1/60 second<br>No automatic call |
| Any resolution<br>SsSeqCalledTbyT | Tick = 1/tick_mode seconds<br>Automatic call |
| (Any resolution \| SS_NOTICK)<br>SsSeqCallTbyT | Tick = 1/tick_mode seconds<br>No automatic call |

1. tick_mode = SS_TICK50

   1/50 second is 1 Tick; the SEQ file will be played at this resolution.

   When the mode specified by SetVideoMode() is MODE_PAL, use the OS Root Counter Management Service RCntCNT3 with this resolution. PAL vertical sync timing (1/50 sec) is 1 Tick; the SEQ file will be played at this resolution. For MODE_NTSC, generate this resolution with the OS Root Counter Management Service RCntCNT2, and interpret and play back the SEQ file. You cannot use RCntCNT2 in programs at any other resolution.

2. tick_mode = SS_TICK60

   1/60 seconds is 1 Tick; the SEQ file will be played at this resolution.

When the mode specified by SetVideoMode() is MODE_NTSC, use the OS Root Counter Management Service RCntCNT3 with this resolution. NTSC vertical sync timing (1/60 sec) is 1 Tick; the SEQ file will be played at this resolution. For MODE_PAL, generate this resolution with the OS Root Counter Management Service RCntCNT2, and interpret play back the SEQ file. You cannot use RCntCNT2 in programs at any other resolution.

3. tick_mode = SS_TICKVSYNC

   Vertical sync timing is 1 Tick; the SEQ file will be played at this resolution.

   When the mode specified by SetVideoMode() is MODE_NTSC, NTSC vertical sync timing (1/60 sec) is the resolution; when the specified mode is MODE_PAL, PAL vertical sync timing (1/50 sec) is the resolution. The SEQ file is interpreted and played back.

   Use the OS Root Counter Management Service RCntCNT3 at both of these resolutions.

4. tick_mode = SS_TICK120

   1/120 seconds are 1 Tick; the SEQ file will be played at this resolution. However, because the OS Root Counter Management Service RCntCNT2 is used at this resolution, it cannot be used by programs at other than this resolution.

5. tick_mode = SS_TICK240

   1/240 seconds is 1 Tick; the SEQ file will be played at this resolution. However, because the OS Root Counter Management Service RCntCNT2 is used at this resolution, it cannot be used by programs at other resolutions.

6. tick_mode = SS_NOTICK

   Vertical retrace timing (1/60 seconds) is 1 Tick; the SEQ file will be played at this resolution. However, because SsSeqCalledTbyT() will not be automatically called, it must be called inside the user program at the vertical retrace timing.

7. tick_mode = Any resolution

   By setting a value between 60 and 240 in the argument, 1 Tick is set to (1/tick_mode), and the SEQ file is interpreted and played at this resolution. However, in this case, because the OS Root Counter Management Service RCntCNT2 is used at this resolution, it cannot be used by programs at other than this resolution.

8. tick_mode = (Any resolution | SS_NOTICK)

   By setting a value between 60 and 240 in the argument, 1 Tick is set at (1/tick_mode), and the SEQ file is interpreted and played at this resolution. However, if SS_NOTICK is specified as "bit or" in an argument, SsSeqCalledTbyT() will not be called automatically, so the user must call SsSeqCalledTbyT() at the timing specified by the program.

### Return value

None.

### Remarks

**See also:** SsStart (), SsSeqCalledTbyT (p. 12-31), SetVideoMode (see libetc).

## SsStart

Starts the sound system.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsStart** (*void*)

### Arguments

None.

### Explanation

Carries out the sound system start process.

When the mode is set to call SsSeqCalledTbyT () automatically by SsSetTickMode (), SsSeqCalledTbyT () is called in each Tick after calling this function.

### Return value

None.

### Remarks

**See also:** SsEnd (p. 12-1), SsSetTickMode (p. 12-62), SsSeqCalledTbyT (p. 12-31).

## SsStart2

Starts the sound system (VSyncCallback version).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.1* | *7/31/96* |

### Syntax

**void SsStart** (void)

### Arguments

None.

### Explanation

Carries out the sound system start process.

When the mode is set to call SsSeqCalledTbyT() automatically by SsSetTickMode(), SsSeqCalledTbyT() is called in each Tick after calling this function.

Set SsSeqCalledTbyT() in VSyncCallback() only when SS_TICK60 on NTSC or SS_TICK50 on PAL is specified in SsSetTickMode(). The setting of SsSeqCalledTbyT() in other Tick modes is the same as SsStart()

### Return value

None.

### Remarks

**See also:** SsStart (), SsEnd (p. 12-8), SsSetTickMode (p. 12-62), SsSeqCalledTbyT (p. 12-31).

## SsUtAllKeyOff

Keys off all voices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsUtAllKeyOff** (**short** *mode*)
**short** *mode*;

### Arguments

| | |
|---|---|
| *mode* | Always 0 |

### Explanation

Forcibly keys off all voices used by libsnd.

### Return value

None.

### Remarks

**See also:**

## SsUtAutoPan

Automatically changes panning.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtAutoPan** (*vc*, *start_pan*, *end_pan*, *delta_time*)
**short** *vc*;
**short** *start_pan*;
**short** *end_pan*;
**short** *delta_time*;

### Arguments

| | |
|---|---|
| *vc* | Voice number (0-23) |
| *start_pan* | Panning change starting value (0-127) |
| *end_pan* | Panning change starting value (0-127) |
| *delta_time* | Change starting time (in units of 1/60 sec, to a maximum of 180 Seconds) (0-10800) |

### Explanation

Linearly changes the panning from *start_pan* to *end_pan* at *delta_time* (1/60 sec increments) for voice *vc*.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtAutoVol

Automatically changes voice volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtAutoVol** (*vc*, *start_vol*, *end_vol*, *delta_time*)
**short** *vc*;
**short** *start_vol*;
**short** *end_vol*;
**short** *delta_time*; ;

### Arguments

| | |
|---|---|
| *vc* | Voice number (0-23) |
| *start_vol* | Volume change starting value (0-127) |
| *end_vol* | Volume change starting value (0-127) |
| *delta_time* | Change starting time (in units of 1/60 sec, to a maximum of 180 Seconds) (0-10800) |

### Explanation

Linearly changes the volume from *start_vol* to *end_vol* at *delta_time* (1/60 sec increments) for voice *vc*.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtChangeADSR

Changes ADSR.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtChangeADSR** (*vc*, *vabId*, *prog*, *old_note*, *adsr1*, *adsr2*)
**short** *vc*;
**short** *vabId*;
**short** *prog*;
**short** *old_note*;
**unsigned short** *adsr1*;
**unsigned short** *adsr2*;

### Arguments

| | |
|---|---|
| *vc* | Voice number (0-23) |
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *prog* | Program number (0-127) |
| *old_note* | Previous pitch specification in half-tone units (note number)(0-127) |
| *adsr1* | ADSR1 |
| *adsr2* | ADSR2 |

### Explanation

Changes the ADSR of the voice.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtChangePitch

Changes the pitch.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtChangePitch** (*voice*, *vabId*, *prog*, *old_note*, *old_fine*, *new_note*, *new_fine*)
**short** *voice*;
**short** *vabId*;
**short** *prog*;
**short** *old_note*;
**short** *old_fine*;
**short** *new_note*;
**short** *new_fine*;

### Arguments

| | |
|---|---|
| *voice* | Voice number (0-23) |
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *prog* | Program number (0-127) |
| *old_note* | Previous pitch specification in semitones (note number) (0-127) |
| *old_fine* | Previous fine pitch specification (100/127 cents) (0-127) |
| *new_note* | New pitch specification in semitones (note number) (0-127) |
| *new_fine* | New fine pitch specification (100/127 cents) (0-127) |

### Explanation

Changes the pitch of the voice.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtPitchBend (p. 12-92), SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtFlush

Executes KeyOn/KeyOff requests that have been queued. (Flushing)

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsUtFlush** (*void*)

### Arguments

None.

### Explanation

Executes KeyOn/KeyOff requests that have been queued.

Normally, flushing is performed by an automatic interrupt of Sound Library (when the mode is set by SsSetTickMode to mode other than SS_NOTICK) or by a clear call of SsSeqCalledTbyT (when the mode is set by SsSetTickMode to SS_NOTICK).

However, if neither of these is used, use this function for flushing.

An interval of at least 1/44100 sec must be inserted before calling this function.

### Return value

None.

### Remarks

**See also:**

## SsUtGetDetVVol

Obtains a detailed value of voice volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtGetDetVVol** (*vc*, **detvoll*, **detvolr*)
**short** *vc*;
**short** **detvoll*;
**short** **detvolr*;

### Arguments

| | |
|---|---|
| *vc* | Voice number (0-23) |
| *detvoll* | Pointer to detailed volume, left (0-16383) |
| *detvolr* | Pointer to detailed volume, right (0-16383) |

### Explanation

Returns the detailed value of the voice volume.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtSetDetVVol (), SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtGetProgAtr

Gets a program attribute table.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtGetProgAtr** (*vabId*, *progNum*, **progatrptr*)
**short** *vabId*;
**short** *progNum*;
**ProgAtr** **progatrptr*;

### Arguments

| | |
|---|---|
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *progNum* | Program number (0-127) |
| *progatrptr* | Pointer to program attribute table |

### Explanation

Specifies a VAB number and a program number, and returns the VAB attribute  table to progatrptr.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtSetProgAtr (p. 12-87), ProgAtr (p. 12-3).

## SsUtGetReverbType

Obtains a reverb type.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsUtGetReverbType** (*void*)

### Arguments

None.

### Explanation

Obtains the current reverb type value.

### Return value

Current reverb type value.

### Remarks

**See also:** SsUtSetReverbType ().

## SsUtGetVabHdr

Returns VAB attribute header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtGetVabHdr** (*vabId*, **vabhdrptr*)
**short** *vabId*;
**VabHdr** **vabhdrptr*;

### Arguments

| | |
|---|---|
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *vabhdrptr* | Pointer to VAB attribute header |

### Explanation

Specifies the VAB number (the return value of SsVabOpenHead()) and returns the VAB attribute header to *vabhdrptr*.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** VabHdr (p. 12-5).

## SsUtGetVagAddr

Returns an SPU buffer address stored by VAG.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.1* | *7/31/96* |

### Syntax

**long SsUtGetVagAddr** (*vabId*, *vagId*)
**short** *vabId*;
**short** *vagId*;

### Arguments

| | |
|---|---|
| *vabId* | VAB data id |
| *vagId* | VAG data id |

### Explanation

Given VAB id (0-15) and VAG id (1-254), this function returns a 32-bit SPU buffer address (as bytes) stored by VAG.

### Return value

Returns an SPU buffer address stored by VAG.

### Remarks

**See also:** SsVabOpenHead (p. 12-108).

## SsUtGetVagAddrFromTone

Returns the SPU buffer address where VAG data is stored.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.3* | *7/31/96* |

### Syntax

**unsigned long SsUtGetVagAddrFromTone** (**short** *vabid*, **short** *progid*, **short** *toneid*)

### Arguments

| | |
|---|---|
| *vabid* | VAB id |
| *progid* | Program number |
| *toneid* | Tone number |

### Explanation

This function returns the address in the sound buffer where the VAG wave form data with the specified VAB id, program number, and tone number are transferred.

### Return value

Address in the sound buffer. If it fails, it returns -1.

### Remarks

**See also:**

## SsUtGetVagAtr

Returns a tone attribute table (VagAtr).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtGetVagAtr** (*vabId*, *progNum*, *toneNum*, **vagatrptr*)
**short** *vabId*;
**short** *progNum*;
**short** *toneNum*;
**VagAtr** **vagatrptr*;

### Arguments

| | |
|---|---|
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *progNum* | Program number (0-127) |
| *toneNum* | Tone number (0-15) |
| *vagatrptr* | Pointer to tone attribute table |

### Explanation

Specifies a VAB number, a program number, and a tone number, and returns a tone attribute table to *vagatrptr*.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtSetVagAtr (), VagAtr (p. 12-6).

## SsUtGetVBaddrInSB

Returns the address inside the sound buffer to which VAB data specified by VAB id has been transferred.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SsUtGetVBaddrInSB** (*vabid*)
**short** *vabid*;

### Arguments

| | |
|---|---|
| *vabid* | VAB id |

### Explanation

Returns the address inside the sound buffer to which VAB data specified by VAB id has been transferred.

### Return value

Address inside the sound buffer. Returns -1 if unsuccessful.

### Remarks

**See also:**

## SsUtGetVVol

Obtains voice volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtGetVVol** (*vc*, **voll*, **volr*)
**short** *vc*;
**short** **voll*;
**short** **volr*;

### Arguments

| | |
|---|---|
| *vc* | Voice number (0-23) |
| *voll* | Pointer to volume, left (0-127) |
| *volr* | Pointer to volume, right (0-127) |

### Explanation

Returns a volume value for a voice.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtSetVVol (), SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtKeyOff

Keys off voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsUtKeyOff** (*voice*, *vabId*, *prog*, *tone*, *note*)
**short** *voice*;
**short** *vabId*;
**short** *prog*;
**short** *tone*;
**short** *note*;

### Arguments

| | |
|---|---|
| *voice* | Voice number (0-23) access number |
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *prog* | Program number (0-127) |
| *tone* | Tone number (0-15) |
| *note* | Pitch specification in half-tone units (note number) (0-127) |

### Explanation

Keys off the voice.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtKeyOffV

Keys off the voice specified by the voice number.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsUtKeyOffV** (*voice*)
**short** *voice*;

### Arguments

| | |
|---|---|
| *voice* | Voice number (0-23) |

### Explanation

Keys off the voice specified by the voice number (0-23).

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtKeyOnV (p. 12-91).

## SsUtKeyOn

Keys on voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsUtKeyOn** (*vabId*, *prog*, *tone*, n*ote*, *fine*, *voll*, *volr*)
**short** *vabId*;
**short** *prog*;
**short** *tone*;
**short** *note*;
**short** *fine*;
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *prog* | Program number (0-127) |
| *tone* | Tone number (0-15) |
| *note* | Pitch specification in semitones (note number) (0-127) |
| *fine* | Detailed pitch specification (100/127 cents) (0-127) |
| *voll* | Volume, left (0-127) |
| *volr* | Volume, right (0-127) |

### Explanation

Keys on the voice specified by the VAB number, the program number (0-127), and the tone number (0-15) at the specified pitch and volume, and returns the allocated voice number.

### Return value

Returns the voice number (0-23) used for KeyOn.

Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtKeyOff (p. 12-88).

## SsUtKeyOnV

Keys on the voice specified by voice number.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsUtKeyOnV** (*voice*, *vabId*, *prog*, *tone*, *note*, *fine*, *voll*, *volr*)
**short** *voice*;
**short** *vabId*;
**short** *prog*;
**short** *tone*;
**short** *note*;
**short** *fine*;
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *voice* | Voice number (0-23) |
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *prog* | Program number (0-127) |
| *tone* | Tone number (0-15) |
| *note* | Pitch specification in semitones (note number) (0-127) |
| *fine* | Detailed pitch specification (100/127 cents) (0-127) |
| *voll* | Volume, left (0-127) |
| *volr* | Volume, right (0-127) |

### Explanation

Keys on the voice specified by the voice number (0-23), the VAB number, the program number (0-127), and the tone number (0-15) at the specified pitch and volume, and returns the allocated voice number.

### Return value

Returns the voice number (0-23) used for KeyOn.

Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtKeyOffV (p. 12-89).

## SsUtPitchBend

Applies a pitch bend.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtPitchBend** (*voice*, *vabId*, *prog*, *note*, *pbend*)
**short** *voice*;
**short** *vabId*;
**short** *prog*;
**short** *note*;
**short** *pbend*;

### Arguments

| | |
|---|---|
| *voice* | Voice number (0-23) |
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *prog* | Program number (0-127) |
| *note* | Pitch specification in half-tone units (note number) (0-127) |
| *pbend* | Pitch-bend value (0-127) |

### Explanation

Applies a pitch bend (0-127, 64:center) to the voice.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtChangePitch (p. 12-67), SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtReverbOff

Turns off Reverb.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsUtReverbOff** (*void*)

### Arguments

None.

### Explanation

Turns off Reverb.

### Return value

None.

### Remarks

**See also:** SsUtReverbOn ().

## SsUtReverbOn

Turns on Reverb.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsUtReverbOn** (*void*)

### Arguments

None.

### Explanation

Turns on Reverb at the Type and Depth. Set by SsUtSetReverbType and SsUtSetReverbDepth.

### Return value

None.

### Remarks

**See also:** SsUtReverbOff (), SsUtSetReverbType (p. 12-100), SsUtSetReverbDepth (p. 12-98).

## SsUtSetDetVVol

Sets a detailed value of voice volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsUtSetDetVVol** (*vc*, *detvoll*, *detvolr*)
**short** *vc*;
**short** *detvoll*;
**short** *detvolr*;

### Arguments

| | |
|---|---|
| *vc* | Voice number (0-23) |
| *detvoll* | Detailed volume, left (0-16383) |
| *detvolr* | Detailed volume, right (0-16383) |

### Explanation

Sets the detailed value of voice volume.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtGetDetVVol (), SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsUtSetProgAtr

Sets a program attribute table.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtSetProgAtr** (*vabId*, *progNum*, **progatrptr*)
**short** *vabId*;
**short** *progNum*;
**ProgAtr** **progatrptr*;

### Arguments

| | |
|---|---|
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpen() |
| *progNum* | Program number (0-127) |
| *progatrptr* | Pointer to program attribute table |

### Explanation

Specifies a VAB number and a program number, and changes the program attribute table, progatrptr.

- Change allowed: mvol, mpan, prior, mode, attr
- Change not allowed: tones, reserved0, reserved 1, reserved 2

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtGetProgAtr (p. 12-73).

## SsUtSetReverbDelay

Sets a Delay volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsUtSetReverbDelay** (*delay*)
**short** *delay*;

### Arguments

| | |
|---|---|
| *delay* | 0-127 |

### Explanation

Sets a delay volume for using Echo and Delay type reverb.

### Return value

None.

### Remarks

**See also:**

## SsUtSetReverbDepth

Sets a reverb depth.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsUtSetReverbDepth** (*ldepth*, *rdepth*)
**short** *ldepth*;
**short** *rdepth*

### Arguments

| | |
|---|---|
| *ldepth* | Left depth (0-127) |
| *rdepth* | RIght depth (0-127) |

### Explanation

*ldepth* 0-127

*rdepth* 0-127

Sets a reverb depth.

### Return value

None.

### Remarks

**See also:** SsUtGetReverbDepth (p. 12-81).

## SsUtSetReverbFeedback

Sets a feedback volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**void SsUtSetReverbFeedback** (*feedback*)
**short** *feedback*;

### Arguments

| | |
|---|---|
| *feedback* | Feedback (0-127) |

### Explanation

*feedback* 0-127

Sets a feedback volume for using Echo and Delay type reverb.

### Return value

None.

### Remarks

**See also:**

## SsUtSetReverbType

Sets reverb type.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsUtSetReverbType** (*type*)
**short** *type*;

### Arguments

| | |
|---|---|
| *type* | Reverb type |

**Table 12–12: Reverb Type Overview (See Sound Delicatessen DSP)**

| Type | Mode | Delay time | Feedback |
|---|---|---|---|
| SPU_REV_TYPE_OFF | off | X | X |
| SPU_REV_TYPE_ROOM | room | X | X |
| SPU_REV_TYPE_STUDIO_A | studio (small) | X | X |
| SPU_REV_TYPE_STUDIO_B | studio (med) | X | X |
| SPU_REV_TYPE_STUDIO_C | studio (big) | X | X |
| SPU_REV_TYPE_HALL | hall | X | X |
| SPU_REV_TYPE_SPACE | space echo | X | X |
| SPU_REV_TYPE_ECHO | echo | O | O |
| SPU_REV_TYPE_DELAY | delay | O | O |
| SPU_REV_TYPE_PIPE | pipe echo | X | X |

### Explanation

Sets reverb type.

When a reverb type is set, reverb depth is automatically set to 0. Because noise will occur as soon as depth is set if data remains in the reverb work area, follow the procedure below.

SsUtSetReverbType (SS_REV...);

SsUtReverbOn();

Wait for several seconds.

SsUtSetReverbDepth (64, 64);

Number and type are as shown in the table above.

### Return value

If setting was correctly performed, the Type number that was set is returned.

If setting was not correctly performed, -1 is returned.

### Remarks

**See also:** SsUtGetReverbType (p. 12-81).

## SsUtSetVabHdr

Sets a VAB attribute header.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtSetVabHdr** (*vabId*, **vabhdrptr*)
**short** *vabId*;
**VabHdr** **vabhdrptr*;

### Arguments

| | |
|---|---|
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpenHead |
| *vabhdrptr* | Pointer to VAB attribute header |

### Explanation

Specifies the VAB number (the return value of SsVabOpenHead()) and changes the VAB attribute header, vabhdrptr.

- Setting allowed: mvol, pan, attr1, attr2 only
- Setting not allowed: form, ver, id, fsize, reserved0, ps, ts, vs, reserved 1

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtGetVabHdr (p. 12-82).

## SsUtSetVagAtr

Sets a tone attribute table (VagAtr).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtSetVagAtr** (*vabId*, *progNum*, *toneNum*, **vagatrptr*)
**short** *vabId*;
**short** *progNum*;
**short** *toneNum*;
**VagAtr** **vagatrptr*;

### Arguments

| | |
|---|---|
| *vabId* | VAB number (0-31) from the return value of the function SsVabOpen() |
| *progNum* | Program number (0-127) |
| *toneNum* | Tone number (0-15) |
| *vagatrptr* | Pointer to tone attribute table |

### Explanation

Specifies a VAB number, a program number, and a tone number, and changes a tone attribute table, *vagatrptr*.

Change allowed: Items in VagAtr that are not listed below.

Change not allowed: prog, vag, reserved1, reserved2, reserved[0-3]

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtGetVagAtr (p. 12-76), VagAtr (p. 12-6).

## SsUtSetVVol

Sets voice volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**short SsUtSetVVol** (*short vc*, *short voll*, *short volr*)
**short** *vc*;
**short** *voll*;
**short** *volr*;

### Arguments

| | |
|---|---|
| *vc* | Voice number (0-23) |
| *voll* | Volume, left (0-127) |
| *volr* | Volume, right (0-127) |

### Explanation

Sets the volume of the voice.

### Return value

Returns 0 if successful. Returns -1 if unsuccessful.

### Remarks

**See also:** SsUtGetVVol (), SsUtKeyOn (p. 12-90), SsUtKeyOnV (p. 12-91), SsVoKeyOn (p. 12-115).

## SsVabClose

Closes VAB data file.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *2.x* | *7/31/96* |

### Syntax

**void SsVabClose** (*vab_id*)
**short** *vab_id*;

### Arguments

| | |
|---|---|
| *vab_id* | VAB data id |

### Explanation

This function closes a VAB data file containing vab_*id*.

### Return value

None.

### Remarks

**See also:** SsVabOpen (p. 12-100).

## SsVabFakeBody

Recognizes sound source data in the sound buffer as the given VAB ID. This function does not perform any transfer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabFakeBody** (*vabid*)
**short** *vabid*;

### Arguments

| | |
|---|---|
| *vabid* | VAB id |

### Explanation

This function rerecognizes sound source data in the sound buffer after SsVabFakeHead has rerecognized a header list on main RAM.

Although this function does perform VAB ID verification, it does not perform the actual transfer. Instead, it sets the internal state of the library to "Transferred to SPU."

It is not necessary to use SsVabTransCompleted after calling this function.

### Return value

VAB Identifying number. Returns -1 if unsuccessful.

### Remarks

**See also:** SsVabFakeHead (p. 12-106), SsVabOpenHeadSticky (p. 12-109), SsVabTransBody (p. 12-110), SsVabTransBodyPartly (p. 12-111).

## SsVabFakeHead

Rerecognizes a sound source header list.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabFakeHead** (**addr*, *vabid*, *sbaddr*)
**unsigned char** **addr*;
**short** *vabid*;
**unsigned long** *sbaddr*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to VH leading address |
| *vabid* | Desired VAB ID. If "-1", the library will make the allocation. |
| *sbaddr* | Address inside the sound buffer, to which VB is being transferred. |

### Explanation

Rerecognizes the sound source header in the main memory, and sets the previously read VH data in the state that can be used by the library again.

Specify a VAB ID for opening. When VAB ID is -1, the function searches for an empty VAB ID (0 - 16) and allocates.

The user must specify the leading address in *sbaddr* for the area inside the sound buffer to which VB is being transferred.

### Return value

VAB Identifying number. Returns -1 if unsuccessful.

### Remarks

**See also:** SsVabFakeBody (), SsUtGetVBaddrInSB (p. 12-80),
SsVabOpenHead (), SsVabOpenHeadSticky (
).

## SsVabOpen

Opens VAB data.

NOTE: This function is no longer recommended for use. Instead, use SsVabOpenHead and SSVabTransBody or SsVabTransfer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabOpen** (**addr*, **vab_header*)
**unsigned char** **addr*;
**VabHdr** **vab_header*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to start address of VAB data in main storage |
| *vab_header* | Pointer to address to VAB header structure corresponding to VAB id |

### Explanation

It analyses the VAB data header which is in the main memory, stores the header value in *vab_header*, and returns the VAB id that identifies the VAB given as the function’s Return value. At the same time, it transmits to the SPU local memory the VAG data group (wave form) data contained in VAB.

### Return value

It is the VAB id which identifies the given VAB. It is -1 in the event of failure.

**Note:**

This function is no longer recommended for use. Instead, use SsVabOpenHead and SsVabTransBody or SsVabTransfer.

### Remarks

**See also:** SsVabClose (p. 12-97), SsVabOpenHead (p. 12-108), SsVabTransBody (p. 12-110).

## SsVabOpenHead

Recognizes a sound source header list.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabOpenHead** (**addr*, *vabid*)
**unsigned char** **addr*;
**short** *vabid*

### Arguments

| | |
|---|---|
| *addr* | Pointer to VAB data starting address |
| *vabid* | VAB ID |

### Explanation

Recognizes a sound source header list in the main memory.

Sets the table in the main memory so that it can be used by the Sound Library. Specify a VAB ID for opening. When VAB ID is -1, the function searches for an empty VAB ID (0 - 15) and allocates it.

### Return value

VAB identification number. Returns -1 if unsuccessful.

### Remarks

**See also:** SsVabTransBody (p. 12-110), SsVabTransBodyPartly (p. 12-111), SsVabOpenHeadSticky (p. 12-109), SsVabTransfer (p. 12-113).

## SsVabOpenHeadSticky

Recognizes a sound source header list. (.VB transfer address specification).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabOpenHeadSticky** (**addr*, *vabid*, *sbaddr*)
**unsigned char** **addr*;
**short** *vabid*;
**unsigned long** *sbaddr*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to leading address of VAB data in the main memory |
| *vabid* | Desired VAB ID or -1 |
| *sbaddr* | Leading address inside the sound buffer to be usedwhen transferring VabBody (.VB) to the sound buffer |

### Explanation

Recognizes a sound source header list in the main memory.

Sets the table in the main memory in the state that is usable by Sound Library. Specify a VAB ID for opening. When VAB ID is -1, the function searches for an empty VAB ID (0-15) and allocates.

Specify for *sbaddr* the leading address inside the sound buffer for transferring VabBody (.VB) to the sound buffer, within the range of 0x1010 to 0x7ffff. When doing so, take .VB size into consideration and specify the address so that the it will not be transferred into the reverb work area.

SsVabTransBody/SsVabTransBodyPartly that is called later transfers VabBody to sbaddr.

When using this function, because consistency cannot be maintained for the sound buffer memory management, SsVabOpenHead will not be able to be used when opening other VAB (.VH). Use this function to open all .VH.

### Return value

VAB identifying number. Returns -1 if unsuccessful.

### Remarks

**See also:** SsVabOpenHead (), SsVabTransBody ().
SsVabTransBodyPartly (), SsVabTransfer (p. 12-113).

## SsVabTransBody

Transfers sound source data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabTransBody** (**addr*, *vabid*)
**unsigned char** **addr*;
**short** *vabid*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to VAB data leading address |
| *vabid* | VAB ID |

### Explanation

After SsVabOpenHead is used for recognizing a header list, SsVabTransBody starts the transfer of the sound source data (VAB body) in the main memory to the SPU local memory.

### Return value

VAB identifying number. Returns -1 if unsuccessful.

### Remarks

**See also:** SsVabOpenHead (), SsVabTransBodyPartly ()
, SsVabOpenHeadSticky (p. 12-109), SsVabTransfer (p. 12-113).

## SsVabTransBodyPartly

Transfers sound source data in segments.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabTransBodyPartly** (**addr*, *bufsize*, *vabid*)
**unsigned char** **addr*;
**unsigned long** *bufsize*;
**short** *vabid*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to starting address of the segment transfer buffer |
| *bufsize* | Buffer size |
| *vabid* | VAB ID |

### Explanation

Starts transfer to the SPU sound buffer of main memory sound source data (VAB body) whose data header list is recognized using SsVabOpenHead().

By continuously calling SsVabTransBodyPartly() while sequentially copying part of the sound source (VAB body) into the area possessing a *bufsize* indicated by *addr*, transfers may be made to a contiguous area within the sound buffer using only a limited area in main memory.

In order to ensure continuity of transfer, you must use SsVabTransCompleted() to verify whether each transfer has been completed, after SsVabTransBodyPartly() has been called.

### Return value

Transfer results return the following values.

**Table 12–13**

| Return value | Status |
|---|---|
| -2 | The size of the sound source data (VAB body) inherited from SsVabOpenHead() has not been completely transferred |
| -1 | Transfer failed |
| vabid | Transfer successful |

### Remarks

**See also:** SsVabOpenHead (), SsVabTransBody ( ), SsVabOpenHeadSticky (p. 12-109), SsVabTransfer (p. 12-113).

## SsVabTransCompleted

Gets VAB data transfer state.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**short SsVabTransCompleted** (*immediateFlag*)
**short** *immediateFlag*;

### Arguments

| | |
|---|---|
| *immediateFlag* | Transfer status recognition flag |

### Explanation

Returns an indication of whether data transfer to SPU local memory has terminated.

*immediateFlag* may be specified with the following values:

**Table 12–14**

| *ImmediateFlag* | Action |
|---|---|
| SS_IMMEDIATE | Immediately returns transfer state |
| SS_WAIT_COMPLETED | Loops until transfer is completed |

### Return value

Returns "1" if the transfer has been completed. Returns "0" if the transfer is ongoing.

### Remarks

**See also:** SsVabOpenHead (p. 12-108), SsVabOpenHeadSticky (p. 12-109), SsVabTransfer (p. 12-113). SsVabTransBody (p. 12-110), SsVabTransBodyPartly (p. 12-111).

## SsVabTransfer

Recognizes and transfers sound source data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**SsVabTransfer (unsigned char** **vh_addr*, **unsigned char** **vb_addr*, **short** *vabid*, **short** *i_flag***)**

### Arguments

| | |
|---|---|
| *vh_addr* | Pointer to starting address of VH data |
| *vb_addr* | Pointer to starting address of VB data |
| *vabid* | VAB ID number |
| *i_flag* | = SS_IMMEDIATE...Immediately returns return value (VAB ID number) |
| | = SS_WAIT_COMPLETED...Waits until transfer is completed |

### Explanation

This function recognizes a sound source header list(VH data) specified by vh_addr and transfers a sound source data(VB data) specified by vb_addr, to the SPU sound buffer. The VAB ID number is specified in the argument "vabid." When "vabid" is -1, the function searches for an empty VAB ID(0-15) and allocates. The "i_flag" determines whether the function should wait until transfer is completed or return immediately after the transfer starts (then checks with SsVabTransCompleted).

### Return value

VAB ID number for successful return.

For error case the following value is returned.

| | |
|---|---|
| -1 | VAB ID cannot be allocated or invalid VH |
| -2 | Invalid VB |
| -3 and lower | Other error |

### Remarks

**See also:** SsVabOpenHead (p. 12-108), SsVabOpenHeadSticky (p. 12-109), SsVabTransBody (p. 12-110), SsVabTransBodyPartly (p. 12-111), SsVabTransCompleted (p. 12-112).

## SsVoKeyOff

Key off.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**long SsVoKeyOff** (*vab_pro*, *pitch*)
**long** *vab_pro*;
**long** *pitch*;

### Arguments

| | |
|---|---|
| *vab_pro* | VAB data id and program number |
| *pitch* | Pitch |

### Explanation

Of the lower 16 bits of *vab_pro*, the upper 8 bits are used for VAB id, and the lower 8 bits specify a program number. Of the lower 16 bits of *pitch*, the upper 8 bits specify a key number in MIDI standard. To specify a finer pitch, specify a key number in the lower 8 bits of pitch in 1/128 semitones.

### Return value

The return value for this function is not useable at this time.

### Remarks

**See also:** SsVoKeyOn (p. 12-115).

## SsVoKeyOn

Key on.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsnd.lib* | *Libsnd.h* | *3.0* | *7/31/96* |

### Syntax

**long SsVoKeyOn** (*vab_pro*, *pitch*, *voll*, *volr*)
**long** *vab_pro*;
**long** *pitch*;
**unsigned short** *voll*;
**unsigned short** *volr*;

### Arguments

| | |
|---|---|
| *vab_pro* | VAB data id and program number |
| *pitch* | Pitch |
| *volL* | Channel volume |
| *volR* | Channel volume |

### Explanation

Of the lower 16 bits of *vab_pro*, the upper 8 bits are used for VAB id, and the lower 8 bits specify a program number. Of the lower 16 bits of *pitch*, the upper 8 bits specify a key number in MIDI standard. To specify a finer pitch, specify a key number in the lower 8 bits of pitch in 1/128 semitone units. The sound specified by *vab_pro* and *pitch* is keyed on at the specified *voll* and *volr*.

### Return value

Returns which voices were keyed on.

AND the return value and SPU_xxCH (xx=0-23)

**Table 12–15**

| Result of AND | Description |
|---|---|
| 0 | Voice not keyed on |
| 1 | Voice keyed on |

### Remarks

**See also:** SsVoKeyOff (p. 12-114).

# Chapter 13: Basic Sound Library
Table of Contents

## SpuCommonAttr

Common attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *2.x* | *7/31/96* |

### Structure

**typedef struct {**
    **unsigned long** *mask*;
    **SpuVolume** *mvol*;
    **SpuVolume** *mvolmode*;
    **SpuVolume** *mvolx*;
    **SpuExtAttr** *cd*;
    **SpuExtAttr** *ext*;
**} SpuCommonAttr;**

### Members

| | |
|---|---|
| *mask* | Set mask |
| *mvol* | Master volume |
| *mvolmode* | Master volume mode |
| *mvolx* | Current master volume |
| *cd* | Cd input attributes |
| *ext* | External digital input attributes |

### Explanation

Used when setting/checking common attributes. The members needed for setting are set as bit values in *mask*.

### Remarks

**See also:** SpuVolume (p. 13-10), SpuExtAttr (p. 13-4), SpuSetCommonAttr (p. 13-67), SpuGetCommonAttr (p. 13-16).

## SpuDecodeData

Decode data.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *2.x* | *7/31/96* |

### Structure

#define SPU_DECODEDATA_SIZE 0x200

**typedef struct {**
  **short** *cd_left*[SPU_DECODEDATA_SIZE];
  **short** *cd_right*[SPU_DECODEDATA_SIZE];
  **short** *voice1*[SPU_DECODEDATA_SIZE];
  **short** *voice3*[SPU_DECODEDATA_SIZE];
**} SpuDecodeData;**

### Members

| | |
|---|---|
| *cd_left* | CD L channel data decoded by SPU |
| *cd_right* | CD R channel data decoded by SPU |
| *voice1* | Voice 1 data decoded by SPU |
| *voice3* | Voice 3 data decoded by SPU |

### Explanation

Used when getting CD-ROM, voice 1 and voice 3 data decoded by the SPU.

The data which can actually be used is each member's first half 0x100 data or second half 0x100 data. This is determined by the return value of SpuReadDecodeData().

### Remarks

**See also:** SpuReadDecodeData (p. 13-61).

## SpuExtAttr

External input attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *2.x* | *7/31/96* |

### Structure

**typedef struct {**
    **SpuVolume** *volume*;
    **long** *reverb*;
    **long** *mix*;
**} SpuExtAttr;**

### Members

| | |
|---|---|
| *volume* | Volume |
| *reverb* | Reverb on/off |
| *mix* | Mixing on/off |

### Explanation

Used when setting/checking CD and external digital input attributes.

### Remarks

**See also:** SpuCommonAttr (p. 13-3), SpuVolume (p. 13-11).

## SpuReverbAttr

Reverb attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct** {
    **unsigned long** *mask*;
    **long** *mode*;
    **SpuVolume** *depth*;
    **long** *delay*;
    **long** *feedback*;
} **SpuReverbAttr**;

### Members

| | |
|---|---|
| *mask* | Set mask |
| *mode* | Reverb mode |
| *depth* | Reverb depth |
| *delay* | DelayTime (ECHO, DELAY only) |
| *feedback* | Feedback (ECHO, DELAY only) |

### Explanation

Used when setting/checking reverb attributes. The members required at setting are set in the mask as bit values.

### Remarks

**See also:** structure SpuVolume (p. 13-11), SpuSetReverbModeParam (p.13-82), SpuGetReverbModeParam (p. 13-25), SpuSetReverbDepth (p. 13-81).

## SpuStEnv

SPU streaming environment attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *2.x* | *7/31/96* |

### Structure

```
typedef struct {
    long size;
    SpuStVoiceAttr voice[24];
} SpuStEnv
```

### Members

| | |
|---|---|
| *size* | Stream buffer size |
| *voice* | Each stream attribute set |

### Explanation

Used in SPU streaming library, streaming environment and each stream attribute setting.

### Remarks

**See also:** SpuStVoiceAttr (p. 13-8), SpuStInit (p. 13-113).

## SpuStVoiceAttr

SPU streaming voice attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Structure

```
typedef struct {
    char status;
    char pad1;
    char pad2;
    char pad3;
    long last_size;
    unsigned long buf_addr;
    unsigned long data_addr;
} SpuStVoiceAttr
```

### Members

| | |
|---|---|
| *status* | Stream status |
| *pad1* | Padding |
| *pad2* | Padding |
| *pad3* | Padding |
| *last_size* | The size of final data transfer (last_size <= (size / 2)) |
| *buf_addr* | The start address of stream buffer |
| *data_addr* | The start address of stream in SPU RAM data in main RAM |

### Explanation

Holds each stream's attributes in the SPU streaming library.

### Remarks

**See also:** SpuStEnv (p. 13-7), SpuStInit (p. 13-113).

## SpuVoiceAttr

Voice attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Structure

```
typedef struct {
    unsigned long voice;
    unsigned long mask;
    SpuVolume volume;
    SpuVolume volmode;
    SpuVolume volumex;
    unsigned short pitch;
    unsigned short note;
    unsigned short sample_note;
    short envx;
    unsigned long addr;
    unsigned long loop_addr;
    long a_mode;
    long s_mode;
    long r_mode;
    unsigned short ar;
    unsigned short dr;
    unsigned short sr;
    unsigned short rr;
    unsigned short sl;
    unsigned short adsr1;
    unsigned short adsr2;
} SpuVoiceAttr;
```

### Members

| | |
|---|---|
| *voice* | Set voice (value is bit string) |
| *mask* | Set attribute bit (value is bit string) |
| *volume* | Volume |
| *volmode* | Volume mode |
| *volumex* | Current volume |
| *pitch* | Interval (set pitch) |
| *note* | Interval (set note) |
| *sample_note* | Interval (set note) |
| *envx* | Current envelope volume value |
| *addr* | Waveform data start address |
| *loop_addr* | Starting address of loop |
| *a_mode* | Attack rate mode |
| *s_mode* | Sustain rate mode |
| *r_mode* | Release rate mode |
| *ar* | Attack rate |
| *dr* | Decay rate |
| *sr* | Sustain rate |
| *rr* | Release rate |
| *sl* | Sustain level |
| *adsr1* | Same value as structure VagAtr adsr1 |
| *adsr2* | Same value as structure VagAtr adsr2 |

### Explanation

Used when setting/checking the attributes of each voice. The voice number is provided/obtained from the voice bit value, and the members needed for setting are set as bit values in the mask.

**Note:**

Constant macro names spelled SPU_ON, SPU_OFF have the same values as and are interchangeable with constant macros used in the program and spelled SpuOn, SpuOff.

### Remarks

**See also:** structure SpuVolume (p. 13-11), SpuSetVoiceAttr (p. 13-92), SpuGetVoiceAttr (p, 13-33), SpuSetKeyOnWithAttr (p. 13-74).

## SpuVolume

Volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *2.x* | *7/31/96* |

### Structure

```
typedef struct {
    short left;
    short right;
} SpuVolume;
```

### Members

| | |
|---|---|
| *left* | L channel value |
| *right* | R channel value |

### Explanation

Used in attributes that require L channel/R channel values when setting/getting each voice.

### Remarks

**See also:** SpuVoiceAttr (p. 13-119), SpuReverbAttr (p. 13-5), SpuExtAttr (p. 13-4), SpuCommonAttr (p. 13-3).

## SpuClearReverbWorkArea

Clears reverb work area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuClearReverbWorkArea** (*rev_mode*)
**long** *rev_mode*;

### Arguments

| | |
|---|---|
| *rev_mode* | Reverb mode |

### Explanation

Clears the area occupied by the reverb work area corresponding to the reverb mode specified by *rev_mode*.

Regardless of whether or not it is reserved at this time, the function checks to see if the area is being used.

This operation uses synchronous DMA transfer, so depending on the reverb mode, some time may be needed.

### Return value

If successful, 0 is returned.

SPU_ERROR is returned if the reverb work area corresponding to the reverb mode set by *rev_mode* is in use, or if the specified reverb mode value is wrong.

### Remarks

**See also:** SpuSetReverbModeParam (p.13-82), SpuReserveReverbWorkArea (p. 13-63), SpuSetReverb (p. 13-80), SpuMalloc (p. 13-55), SpuMallocWithStartAddr (p. 13-56).

## SpuFlush*

Flushes queued events.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**unsigned long**
**SpuFlush (unsigned long** *ev***)**

### Arguments

| | |
|---|---|
| *ev* | Event to be flushed |

### Explanation

This function flushes a queued event.

Set ev with bitwise inclusive ORed events to be flushed;

| | |
|---|---|
| SPU_EVENT_KEY | Key ON/OFF |
| SPU_EVENT_PITCHLFO | Pitch LFO Voice Set |
| SPU_EVENT_NOISE | Noise Voice Set |
| SPU_EVENT_REVERB | Reverb Voice Set |

When ev is set to SPU_EVENT_ALL, all events will be flushed.

### Return value

Bitwise inclusive ORed value of the flushed event(s).

### Remarks

**See also:** SpuSetEnv(), SpuSetKey(), SpuSetKeyOnWithAttr(), SpuSetPitchLFOVoice(), SpuSetNoiseVoice(), SpuSetReverbVoice().

## SpuFree

Releases area allocated in the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuFree** (*addr*)
**unsigned long** *addr*;

### Arguments

| | |
|---|---|
| *addr* | Start address of allocated area (in bytes) |

### Explanation

Releases area allocated in the sound buffer as indicated by the start address *addr*, and deletes that area's information from the management table.

### Return value

None.

### Remarks

**See also:** SpuInitMalloc (p. 13-52), SpuMalloc (p. 13-55), SpuMallocWithStartAddr (p.13-56).

## SpuGetAllKeysStatus

Determines key on/off for voices in the designated range.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuGetAllKeysStatus** (**status*)
**char** **status*[24];

### Arguments

| | |
|---|---|
| *status*[24] | Pointer to the result of checking a voice |

### Explanation

Checks key on/key off and envelope status of all voices; checks actual key on/key off.

An error may result if multiple envelopes are set, and if volume goes to 0 in the course of changing envelope status.

The current key on/key off and envelope status of each voice is returned to status[24].

**Table 13–1**

| Value | Status |
|---|---|
| SPU_ON | Key on status |
| | Not turned off by SpuSetKey |
| | Envelope not 0 |
| SPU_ON_ENV_OFF | Key on status |
| | Not turned off by SpuSetKey |
| | Envelope 0 |
| SPU_OFF_ENV_ON | Key off status |
| | Turned off by SpuSetKey |
| | Envelope not 0 |
| SPU_OFF | Key off status |
| | Turned off by SpuSetKey |
| | Envelope 0 |

### Return value

None

### Remarks

**See also:** SpuSetKey (p. 13-73), SpuGetKeyStatus (p. 13-19), SpuRGetAllKeysStatus (p.13-64).

## SpuGetCommonAttr

Checks attributes common to all voices (infrequent change requests).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuGetCommonAttr** (**attr*)
**SpuCommonAttr** **attr;*

### Arguments

| | |
|---|---|
| *attr* | Pointer to attributes common to all voices |

### Explanation

Returns attributes common to all voices in *attr*. See SpuSetCommonAttr() for details.

### Return value

None.

### Remarks

**See also:** SpuSetCommonAttr (p. 13-67), SpuCommonAttr (p. 13-3).

## SpuGetIRQ

Checks status of interrupt request on/off.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuGetIRQ** (void)

### Arguments

None.

### Explanation

Checks status of interrupt request on/off.

### Return value

Currently set value.

| | |
|---|---|
| SPU_ON | Interrupt request is set |
| SPU_OFF | Interrupt request is not set |

### Remarks

**See also:** SpuSetIRQ ().

## SpuGetIRQAddr

Checks interrupt request address.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuGetIRQAddr (void)**

### Arguments

None.

### Explanation

Returns interrupt request address value.

### Return value

Currently set address value

### Remarks

**See also:** SpuSetIRQAddr (p.13-71), SpuSetIRQ (p.13-70).

## SpuGetKeyStatus

Checks key on/key off status for specified voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuGetKeyStatus** (*voice_bit*)
**unsigned long** *voice_bit;*

### Arguments

| | |
|---|---|
| *voice_bit* | Checked voice |

### Explanation

Checks key on/key off and  envelope status of specified voices; checks actual key on/key off.

Explicitly specify the voices targeted in voice_bit by ORing together SPU_0CH-SPU_23CH. 1 function call gets the attributes of only 1 voice, so, in the case of multiple specifications, the smallest voice number specified is selected.

An error may result if multiple envelopes are set, and if volume goes to 0 in the course of changing envelope status.

### Return value

If successful, the current key on/key off status and envelope status of the specified voice are returned. (See the table below.) If the specified voice is incorrect, SpuGetKeyStatus() returns -1.

**Table 13–2**

| Value | Status |
|---|---|
| SPU_ON | Key on status |
| | Not turned off by SpuSetKey |
| | Envelope not 0 |
| SPU_ON_ENV_OFF | Key on status |
| | Not turned off by SpuSetKey |
| | Envelope 0 |
| SPU_OFF_ENV_ON | Key off status |
| | Turned off by SpuSetKey |
| | Envelope not 0 |
| SPU_OFF | Key off status |
| | Turned off by SpuSetKey |
| | Envelope 0 |

### Remarks

**See also:** SpuSetKey (p.13-73), SpuGetAllKeysStatus (p.13-15).

## SpuGetMute

Checks status of sound muting.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuGetMute (void)**

### Arguments

None.

### Explanation

Checks current sound mute on/off status.

### Return value

Currently set value (SPU_ON/SPU_OFF)

**Table 13–3**

| Value | Description |
|---|---|
| SPU_ON | Mute on |
| SPU_OFF | Mute off |

### Remarks

**See also:** SpuSetMute (p. 13-76).

## SpuGetNoiseClock

Checks noise source clock.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuGetNoiseClock (void)**

### Arguments

None.

### Explanation

Returns the value of noise source clock.

### Return value

Currently set noise source clock value.

### Remarks

**See also:** SpuSetNoiseClock (p.13-77).

## SpuGetNoiseVoice

Checks noise source ON/OFF for each voice

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuGetNoiseVoice (void)**

### Arguments

None.

### Explanation

Checks current status of noise source ON/OFF for each voice.

### Return value

Returns the noise source ON/OFF value of the current voice. OR together SPU_0CH-SPU_23CH.

Distinguishes the noise source ON/OFF value by ANDing the return value and SPU_xxCH(xx=0~23).

**Table 13–4**

| Result of AND | Description |
|---|---|
| 0 | Noise source off |
| Other than 0 | Noise source on |

### Remarks

**See also:** SpuSetNoiseClock (p. 13-77), SpuSetNoiseVoice (p. 13-78).

## SpuGetPitchLFOVoice

Checks pitch LFO ON/OFF for each voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuGetPitchLFOVoice (void)**

### Arguments

None.

### Explanation

Checks current status of pitch LFO ON/OFF for each voice.

### Return value

Returns the pitch LFO ON/OFF value of the current voice. OR together SPU_0CH-SPU_23CH.

Distinguishes the pitch LFO ON/OFF value by ANDing the return value and SPU_xxCH(xx=0~23).

Table 13–5

| Result of AND | Description |
|---|---|
| 0 | Pitch LFO off |
| Other than 0 | Pitch LFO on |

### Remarks

**See also:** SpuSetPitchLFOVoice (p.13-79).

## SpuGetReverb

Checks reverb status.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuGetReverb (void)**

### Arguments

None.

### Explanation

Checks current reverb ON/OFF status.

### Return value

Set value (SPU_ON/SPU_OFF).

**Table 13–6**

| Value | Description |
|---|---|
| SPU_ON | Reverb on |
| SPU_OFF | Reverb off |

### Remarks

**See also:** SpuSetReverb (p.13-80).

## SpuGetReverbModeParam

Checks reverb mode and parameters.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuGetReverbModeParam** (**attr*)
**SpuReverbAttr** **attr*;

### Arguments

| | |
|---|---|
| *attr* | Pointer to reverb attributes |

### Explanation

Gets currently set reverb mode and parameters.

For details see SpuSetReverbModeParam().

### Return value

None.

### Remarks

**See also:** SpuSetReverbModeParam (p.13-82), SpuReverbAttr (p. 13-5).

## SpuGetReverbVoice

Checks reverb ON/OFF for each voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuGetReverbVoice (void)**

### Arguments

None.

### Explanation

Checks current reverb ON/OFF status for each voice.

### Return value

Returns the reverb ON/OFF value of the current voice. OR together SPU_0CH-SPU_23CH.

Distinguishes the noise source ON/OFF value by ANDing the return value and SPU_xxCH(xx=0~23).

**Table 13–7**

| Result of AND | Description |
|---|---|
| 0 | Reverb off |
| Other than 0 | Reverb on |

### Remarks

**See also:** SpuSetReverbVoice (p.13-84).

## SpuGetTransferMode

Checks sound buffer transfer mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuGetTransferMode** (void)

### Arguments

None.

### Explanation

Returns currently set value of the transfer mode when transferring from main memory to the sound buffer.

### Return value

Current setting of transfer mode

| | |
|---|---|
| SPU_TRANSFER_BY_DMA | DMA transfer setting |
| SPU_TRANSFER_BY_IO | I/O transfer setting |

### Remarks

**See also:** SpuSetTransferMode (p.13-86), SpuWrite (p. 13-121).

## SpuGetTransferStartAddr

Checks sound buffer transfer destination/transfer source start address.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuGetTransferStartAddr (void)**

### Arguments

None.

### Explanation

Returns currently set value for start address when transferring from main memory to the sound buffer, and from the sound buffer to main memory.

### Return value

Currently set sound buffer starting address value.

### Remarks

**See also:** SpuSetTransferStartAddr (p.13-87), SpuWrite (p. 13-121), SpuRead (p.13-60).

## SpuGetVoiceADSR*

Gets ADSR.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceADSR** *(***int** *voiceNum,* **unsigned short** **AR,* **unsigned short** **DR*
**unsigned short** **SR,* **unsigned short ****RR,*
**unsigned short** **SL)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |
| *DR* | ADSR decay rate |
| *SR* | ADSR sustain rate |
| *RR* | ADSR release rate |
| *SL* | ADSR sustain level |

### Explanation

This function obtains each ADSR attribute used in the voice, equivalent to the process to obtain the values for SpuVoiceAttr members, AR, DR, SR, RR, and SL using SpuGetVoiceAttr function.

The value obtained are valid only when the attack, sustain, and release rate are set to the mode as below:

```
---------------------+-----------------------------------

Attack Rate Mode | SPU_VOICE_LINEARIncN (Linear Increase)

Sustain Rate Mode | SPU_VOICE_LINEARDecN (Linear Decrease)

Release_Rate_Mode | SPU_VOICE_LINEARDecN_(Linear Decrease)

---------------------+-----------------------------------
```

For other mode, the obtained values are undefined. If you want to obtain multiple Rate Mode at the same time, use SpuSetVoiceADSRAttr.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(p. 13-33), SpuNGetVoiceAttr(p. 13-57), SpuGetVoiceAR(p. 13-31), SpuGetVoiceDR(p. 13-34), SpuGetVoiceSR(p. 13-44), SpuGetVoiceRR(p. 13-40), SpuGetVoiceSL(p. 13-43).

## SpuGetVoiceADSRAttr*

Gets ADSR and each mode..

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceADSRAttr** *(***int** *voiceNum,* **unsigned short** **AR,* **unsigned short** **DR*
**unsigned short** **SR,* **unsigned short** **RR,* **unsigned short** **SL*
**long** **ARmode,* **long** **SRmode,* **long** **RRmode)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |
| *DR* | ADSR decay rate |
| *SR* | ADSR sustain rate |
| *RR* | ADSR release rate |
| *SL* | ADSR sustain level |
| *ARmode* | ADSR attack rate mode |
| *SRmode* | ADSR sustain rate mode |
| *RRmode* | ADSR release rate mode |

### Explanaton

This function obtains each ADSR attribute used in the voice, equivalent to the process to obtain the values for SpuVoiceAttr members, AR, DR, SR, RR, SL, ARmode, SLmode, and RRmode using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceADSR(), SpuGetVoiceAR(), SpuGetVoiceDR(), SpuGetVoiceSR(), SpuGetVoiceRR(), SpuGetVoiceSL(), SpuGetVoiceARAttr(), SpuGetVoiceSRAttr(), SpuGetVoiceRRAttr().

## SpuGetVoiceAR*

Gets ADSR attack rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceAR** *(***int** *voiceNum,* **unsigned short** **AR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |

### Explanation

This function obtains ADSR attack rate used in voice. This function obtains voice volume, equivalent to the process to obtain the value for SpuVoiceAttr member, are using SpuGetVoiceAttr function.

The value obtained is valid only when ADSR attack rate mode is set to SPU_VOICE_LINEARIncN (Linear Increase). For other ADSR attack rate mode the value is undefined.

When both ADSR attack rate volume and ADSR attack rate mode need to be obtained at the same time, use SpuGetVoiceARAttr.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceARAttr().

## SpuGetVoiceARAttr*

Gets ADSR attack rate / attack rate mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceAR**Attr *(***int** *voiceNum,* **unsigned short** **AR,* **long** **ARmode)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |
| *ARmode`* | ADSR attack rate mode |

### Explanation

This function obtains ADSR attack rate / ADSR attack rate mode used in voice, equivalent to the process to obtain the value for SpuVoiceAttr members, AR and ARmode using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNSGetVoiceAttr(), SpuGetVoiceAR().

## SpuGetVoiceAttr

Checks voice attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuGetVoiceAttr** (**attr*)
**SpuVoiceAttr** **attr;*

### Arguments

| | |
|---|---|
| *attr* | Pointer to voice attributes |

### Explanation

Checks voice attributes.

Explicitly set the single voice (SPU_0CH, SPU_1CH, ... SPU_23CH) checked by *attr*.voice. All the attribute structure members are returned except mask. See SpuSetVoiceAttr() for the details of these attributes.

### Return value

None. (The argument *attr* is the return value.)

### Remarks

**See also:** SpuSetVoiceAttr (p.), SpuRSetVoiceAttr, SpuSetKey (p.), SpuSetKeyOnWithAttr (p.), SpuVoiceAttr (p. 13-6).

## SpuGetVoiceDR*

Gets ADSR decay rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceDR** *(***int** *voiceNum,* **unsigned short** **DR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *DR* | ADSR decay rate |

### Explanation

This function obtains ADSR decay rate used in voice,equivalent to the process to obtain the value for SpuVoiceAttr member, DR using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr().

## SpuGetVoiceEnvelope*

Gets current envelope value.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceEnvelope** *(***int** *voiceNum,* **short** **envx)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *envx* | Current envelope value |

### Explanation

This function obtains the current voice envelope value, equivalent to the process to obtain the value for SpuVoiceAttr envx, using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr().

## SpuGetVoiceEnvelopeAttr*

Gets current voice envelope value and key ON/OFF status.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceEnvelopeAttr** *(***int** *voiceNum,* **long** **keyStat,* **short** **envx)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *keyStat* | Status of voice envelope and key ON/OFF |
| *envx* | Current envelope value |

### Explanation

This function obtains the current voice envelope value and voice key ON/OFF and envelope status.

Refer to SpuGetVoiceAttr for values that can be specified in  keystat, the key ON/OFF and envelope status.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceEnvelope(), SpuSetKey(), SpuGetAllKeysStatus(), SpuRGetAllKeysStatus().

## SpuGetVoiceLoopStartAddr*

Gets loop start address of waveform data in the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceLoopStartAddr** *(***int** *voiceNum,* **unsigned long** **loopStartAddr)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *loopStartAddr* | Loop start address |

### Explanation

This function obtains loop start address of waveform data in the sound buffer, equivalent to the process to obtain the value for SpuVoiceAttr loop_addr, using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetTransferStartAddr().

## SpuGetVoiceNote*

Gets interval (note specification).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceNote** *(***int** *voiceNum,* **unsigned short** **note)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *note* | Interval (note specification) |

### Explanation

This function obtains Voice Interval (Note Specification), equivalent to the process to obtain the value for SpuVoiceAttr member, note using SpuGetVoiceAttr function.

Thus prior to call SpuSetVoiceNote, SpuSetVoiceAttr

SPU_VOICE_SAMPLE_NOTE

or the waveform data sample note feature for voice must be set.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceSampleNote(), SpuGetVoiceSampleNote().

## SpuGetVoicePitch*

Gets interval (pitch specification).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoicePitch** *(***int** *voiceNum,* **unsigned short** **pitch)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *pitch* | Interval (pitch specification) |

### Explanation

This function obtains voice interval (pitch specification), equivalent to the process to obtain the value for SpuVoiceAttr member, pitch using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr().

## SpuGetVoiceRR*

Gets ADSR release rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceRR** *(***int** *voiceNum,* **unsigned short** **RR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *RR* | ADSR release rate |

### Explanation

This function obtains ADSR release rate in voice equivalent to, equivalent to the process to obtain the value for SpuVoiceAttr member,    rr using SpuGetVoiceAttr function.

The value obtained is valid only when ADSR release rate mode is set to SPU_VOICE_LINEARDecN (Linear Decrease mode).

For other ADSR release rate mode, the value is undefined. If you want to obtain both ADSR release rate and ADSR release rate mode at the same time, use SpuGetVoiceRRAttr.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceRRAttr().

## SpuGetVoiceRRAttr*

Gets ADSR release rate / release rate mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceRRAttr** *(***int** *voiceNum,* **unsigned short** **RR,* **long** **RRmode)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *RR* | ADSR release rate |
| *RRmode* | ADSR release rate mode |

### Explanation

This function obtains ADSR release rate / ADSR release rate mode used in voice, equivalent to the process to obtain the value for SpuVoiceAttr members, RR and RRmode using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceRR().

## SpuGetVoiceSampleNote*

Gets waveform data sample note.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceSampleNote** *(***int** *voiceNum,* **unsigned short** **sampleNote)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *sampleNote* | Sets waveform data sample note |

### Explanation

This function obtains waveform data sample note, equivalent to the process to obtain the value for SpuVoiceAttr member, sample_note using SpuGetVoiceAttr function..

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceNote().

## SpuGetVoiceSL*

Gets ADSR sustain level.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceSL** *(***int** *voiceNum,* **unsigned short** **SL)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *SL* | ADSR sustain level |

### Explanation

This function obtains ADSR sustain level. equivalent to  the process to obtain the value for SpuVoiceAttr member, SL using SpuGetVoiceAttr function..

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceRRAttr().

## SpuGetVoiceSR*

Gets ADSR sustain rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceSR** *(***int** *voiceNum,* **unsigned short** **SR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *SR* | ADSR sustain rate |

### Explanation

This function obtains ADSR sustain rate in voice equivalent to,    equivalent to the process to obtain the value for SpuVoiceAttr member, SR using SpuGetVoiceAttr function.

The value obtained is valid only when ADSR sustain rate mode is set to SPU_VOICE_LINEARDecN (Linear Decrease mode).

For other ADSR sustain rate mode, the value is undefined. If you want to obtain both ADSR sustain rate and ADSR sustain rate mode at the same time, use SpuGetVoiceSRAttr.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceSRAttr().

## SpuGetVoiceSRAttr*

Gets ADSR sustain rate / sustain rate mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceSRAttr** *(***int** *voiceNum,* **unsigned short** **SR,* **long** *SRmode*)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *SR* | ADSR sustain rate |
| *SRmode* | ADSR sustain rate mode |

### Explanation

This function obtains ADSR sustain rate / ADSR sustain rate mode used in voice, equivalent to the process to obtain the value for SpuVoiceAttr members, SR and SRmode using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceSR().

## SpuGetVoiceStartAddr*

Gets start address of waveform data in the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceStartAddr** *(***int** *voiceNum,* **unsigned long** **startAddr)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *startAddr* | Waveform data start address |

### Explanation

This function obtains start address of waveform data in the sound buffer, equivalent to the process to obtain the value for SpuVoiceAttr member, addr using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetTransferStartAddr().

## SpuGetVoiceVolume*

Gets voice volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceVolume** *(***int** *voiceNum,* **short** **volumeL,* **short** **volumeR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice Number (0 - 23) |
| *volumeL* | Volume (Left) |
| *volumeR* | Volume (Right) |

### Explanation

This function obtains voice volume, equivalent to the process to obtain the value for SpuVoiceAttr member, volume using SpuGetVoiceAttr function.

The value obtained is valid only when the volume mode is set to "Direct Mode". For other volume mode, the value is undefined.

When the volume mode is not "Direct Mode" or both volume and volume mode need to be obtained at the same time, use SpuGetVoiceVolumeAttr.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceVolumeAttr().

## SpuGetVoiceVolumeAttr*

Gets voice volume/volume mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceVolumeAttr** *(***int** *voiceNum,* **short** **volumeL,* **short** **volumeR,* **short** **volModeL,* **short** **volModeR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice Number (0 - 23) |
| *volumeL* | Volume (Left) |
| *volumeR* | Volume (Right) |
| *volModeL* | Volume mode (Left) |
| *volModeR* | Volume mode (Right) |

### Explanation

This function obtains voice volume / volume mode, equivalent to the process to obtain the value for SpuVoiceAttr members, volume and volumed using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceVolume().

## SpuGetVoiceVolumeX*

Gets current voice volume.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuGetVoiceVolumeX** *(***int** *voiceNum,* **short** **volumeL,* **short** **volumeR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice Number (0 - 23) |
| *volumeXL* | Current volume (Left) |
| *volumeXR* | Current volume (Right) |

### Explanation

This functions obtains current voice volume. This function obtains voice volume, equivalent to the process to obtain the value for SpuVoiceAttr member, volumex using SpuGetVoiceAttr function.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceVolume(), SpuGetVoiceVolumeAttr().

## SpuInit

SPU initialization.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.1* | *7/31/96* |

### Syntax

**void SpuInit** (void)

### Arguments

None.

### Explanation

Initializes SPU. Called only once within the program. After initialization, SPU may have the following states.

- Master volume is 0 for both L/R
- Reverb is off
- Reverb work area is not reserved
- Reverb depth is 0 for both L/R
- Reverb volume is 0 for both L/R
- Sound buffer transfer mode is DMA transfer
- For all voices:
  Key off
  Pitch LFO function not set
  Noise function not set
  Reverb function not set
- CD input volume is 0 for both L/R
- External digital input volume is 0 for both L/R
- DMA transfer initialization set

The status of the sound buffer is indeterminate after initialization.

### Return value

None.

### Remarks

**See also:** SpuInitHot (p.), SpuStart (p.), SpuQuit (p.).

## SpuInitHot

Spu initialization (hot reset), preserves sound buffer status.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.1* | *7/31/96* |

### Syntax

**void SpuInitHot** (void)

### Arguments

None.

### Explanation

Initializes SPU. Call SpuInitHot() when you initialize the sound system and want to preserve the sound buffer status in a child process.

After initialization, status is as follows.

- L/R main volume are both 0
- Reverb is off
- Reverb work area is not reserved
- L/R reverb depth are both 0
- L/R reverb volume are both 0
- Transfer to sound buffer is DMA mode
- All voices:
  Key off
  Reverb functionality not yet set
- Sets DMA transfer initialization.

Sound buffer status is preserved after initialization, though not through a hardware reset.

### Return value

None.

### Remarks

**See also:** SpuInit (p.), SpuStart (p.), SpuQuit (p.).

## SpuInitMalloc

Initializes the sound buffer memory management mechanism.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuInitMalloc**(*num*, **top*)
**long** *num*;
**char** **top*;

### Arguments

| | |
|---|---|
| *num* | Maximum number of times memory is allocated |
| *top* | Pointer to the start address of the area storing management table |

### Explanation

Performs initialization in order to divide the sound buffer into *num* number of areas and manage them. The individual *num* memory management blocks used by each request are allocated in the area provided by *top*. The size of the area must be as follows:

(SPU_MALLOC_RECSIZ • (num + 1)) bytes

### Return value

Returns the number of memory management blocks allocated.

### Remarks

When creating memory management blocks to be used by 10 SpuMalloc() calls, SpuInitMalloc() is called as follows:

```
char rec[SPU_MALLOC_RECSIZ * (10 + 1)];
SpuInitMalloc (10,         /*10 SpuMalloc calls can be used*/
   rec);                   /*memory management block*/
```

**See also:** malloc (See libmath), SpuMalloc (p.), SpuMallocWithStartAddr (p.), SpuFree (p. 13-12).

## SpuIsReverbWorkAreaReserved

Checks to see if reverb work area is reserved/Checks to see if reverb work area can be reserved.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuIsReverbWorkAreaReserved**(*on_off*)
**long** *on_off*;

### Arguments

| | |
|---|---|
| *on_off* | Contents of the checking process |

### Explanation

Checks to see if the reverb work area corresponding to the current reverb mode is reserved, or checks to see if it can be reserved.

*on_off* specifies which action is performed. These settings are explained below.

**Table 13–8**

| Value | Description |
|---|---|
| SPU_DIAG | Checks to see if reverb work area can be reserved |
| SPU_CHECK | Checks reverb work area reserve status |

a) SPU_DIAG

Using sound buffer memory management mechanism information, SPU_DIAG checks to see whether or not the reverb work area is an area allocated by SpuMalloc()/SpuMallocWithStartAddr(). If it can be reserved, SPU_ON is returned. If it cannot be reserved, SPU_OFF is returned.

b) SPU_CHECK

Returns current reverb work area reserve status.

### Return value

When *on_off* is SPU_DIAG, if the reverb work area can be reserved, SPU_ON is returned. If it cannot be reserved, SPU_OFF is returned.

When *on_off* is SPU_CHECK, if the reverb work area is reserved, SPU_ON is returned. If it is not reserved, SPU_OFF is returned.

### Remarks

**See also:** SpuReserveReverbWorkArea (p.), SpuSetReverbModeParam (p.), SpuSetReverb (p.), SpuMalloc (p.), SpuMallocWithStartAddr (p.).

## SpuIsTransferCompleted

Checks completion of transfer to the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuIsTransferCompleted** (*flag*)
**long** *flag;*

### Arguments

*flag* Check flag

### Explanation

Checks whether transfer is completed.

Flag values may be specified as follows.

**Table 13–9**

| Value | Description |
|---|---|
| SPU_TRANSFER_WAIT | Wait until transfer ends |
| SPU_TRANSFER_PEEK result | Check whether transfer has ended return |
| SPU_TRANSFER_GLANCE | Same as SPU_TRANSFER_PEEK |

SpuIsTransferCompleted is not functional when, using SpuSetTransferCallback, a callback function is set and started at the completion of DMA transfer.

### Return value

Returns the status of transfer completion.

| | |
|---|---|
| 1 | transfer completed |
| 0 | transfer not completed. |

If flag = SPU_TRANSFER_WAIT, wait until transfer ends and always return 1.

If transfer mode is "I/O transfer", 1 is returned immediately.

SpuIsTransferCompleted returns 1 when, using SpuSetTransferCallback, a callback function is set and started at the completion of DMA transfer.

### Remarks

**See also:** SpuWrite (p. 13-121), SpuRead (p. 13-60).

## SpuMalloc

Allocates an area in the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuMalloc** (*size*)
**long** *size*;

### Arguments

*size* Size of area allocated (in bytes)

### Explanation

Allocate an area of *size* bytes in the sound buffer.

The return value is the address of the start of the allocated area and must be greater than 0x100f. When this value is set by an argument of SpuSetTransferStartAddr(), and the transfer start address is set, SpuWrite() transfers waveform data.

The following states cause failure in allocation.

- The requested size cannot be continuously allocated.
- There is an area which satisfies the requested size, but that area is part or all of a reverb work area allocated by SpuReserveReverbWorkArea(), and essentially cannot be allocated.

### Return value

If allocation is successful, the starting address of the allocated area is returned.

If unsuccessful, -1 is returned.

### Remarks

**See also:** SpuInitMalloc (p. 13-52), SpuMallocWithStartAddr (p. 13-56), SpuFree (p. 13-14), SpuSetTransferStartAddr (p. 13-87), SpuWrite (p. 13-121), SpuReserveReverbWorkArea (p. 13-63), SpuSetReverb (p. 13-80), SpuSetReverbModeParam (p. 13-82).

## SpuMallocWithStartAddr

Allocates an area from a specified start address in sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuMallocWithStartAddr** (*addr*, *size*)
**unsigned** *long addr*;
**long** *size*;

### Arguments

| | |
|---|---|
| *addr* | Allocated area starting address (in bytes) |
| *size* | Size of allocated area (in bytes) |

### Explanation

Allocates an area in the sound buffer of *size* bytes starting from the start address addr.

The allocatable area is 0x01010 - 0x7ffff.

If that address is in an area already allocated, an area of *size* bytes, starting from the nearest empty area after the *addr* area, is allocated.

The following states cause failure in allocation.

- The requested size cannot be continuously allocated.
- There is an area which satisfies the requested size, but that area is part or all of a reverb work area allocated by SpuReserveReverbWorkArea(), and essentially cannot be allocated.

### Return value

If allocation is successful, the starting address of the allocated area is returned. If unsuccessful, -1 is returned.

### Remarks

**See also:** SpuInitMalloc (p. 13-52), SpuMalloc (p.), SpuFree (p. 13-14), SpuSetTransferStartAddr (p. 13-87), SpuWrite (p. 13-121), SpuReserveReverbWorkArea (p. 13-63), SpuSetReverb (p. 13-80), SpuSetReverbModeParam (p. 13-82).

## SpuNGetVoiceAttr*

Gets voice attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuNGetVoiceAttr** *(***int** *voiceNum,* **SpuVoiceAttr** **attr)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *attr* | Voice attribute |

### Explanation

This function obtains voice attribute. Set voice number to be obtained explicitly into voiceNum.

All attributes except "mask" will be returned for "attr" structrue members.

Refer to SpuSetVoiceAttr for detail of each attribute.

### Return value

None.

### Remarks

**See also:** SpuGetVoiceAttr(), SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuRSetVoiceAttr(), SpuSetKey(), SpuSetKeyOnWithAttr(), SpuGetVoiceVolume(), SpuGetVoiceVolumeAttr(), SpuGetVoiceVolumeX(), SpuGetVoicePitch(), SpuGetVoiceNote(), SpuGetVoiceSampleNote(), SpuGetVoiceEnvelope(), SpuGetVoiceStartAddr(), SpuGetVoiceLoopStartAddr(), SpuGetVoiceAR(), SpuGetVoiceDR(), SpuGetVoiceSR(), SpuGetVoiceRR(), SpuGetVoiceSL(), SpuGetVoiceARAttr(),SpuGetVoiceSRAttr(), SpuGetVoiceRRAttr(), SpuGetVoiceADSR(), SpuGetVoiceADSRAttr(), SpuGetVoiceEnvelopeAttr().

## SpuNSetVoiceAttr*

Sets voice attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuNSetVoiceAttr** *(***int** *voiceNum,* **SpuVoiceAttr** **attr)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *attr* | Voice attribute |

### Explanation

This function sets the voice attribute. Set voice number to be obtained explicitly into voiceNum.

Set attr.mask with bitwise inclusive ORed attributes;

| | |
|---|---|
| SPU_VOICE_VOLL | Volume (left) |
| SPU_VOICE_VOLR | Volume (right) |
| SPU_VOICE_VOLMODEL | Volume mode (left) |
| SPU_VOICE_VOLMODER | Volume mode (right) |
| SPU_VOICE_PITCH | Interval (pitch specification) |
| SPU_VOICE_NOTE | Interval (note specification) |
| SPU_VOICE_SAMPLE_NOTE | Waveform data sample note |
| SPU_VOICE_WDSA | Waveform data start address |
| SPU_VOICE_ADSR_AMODE | ADSR attack rate mode |
| SPU_VOICE_ADSR_SMODE | ADSR sustain rate mode |
| SPU_VOICE_ADSR_RMODE | ADSR release rate mode |
| SPU_VOICE_ADSR_AR | ADSR attack rate |
| SPU_VOICE_ADSR_DR | ADSR decay rate |
| SPU_VOICE_ADSR_SR | ADSR sustain rate |
| SPU_VOICE_ADSR_RR | ADSR release rate |
| SPU_VOICE_ADSR_SL | ADSR sustain level |
| SPU_VOICE_ADSR_ADSR1 | ADSR adsr1 for `VagAtr' |
| SPU_VOICE_ADSR_ADSR2 | ADSR adsr2 for `VagAtr' |
| SPU_VOICE_LSAX | Loop start address |

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(),SpuRSetVoiceAttr(), SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuSetKey(), SpuSetKeyOnWithAttr(), SpuSetVoiceVolume(), SpuSetVoiceVolumeAttr(), SpuSetVoicePitch(), SpuSetVoiceNote(), SpuSetVoiceSampleNote(), SpuSetVoiceStartAddr(), SpuSetVoiceLoopStartAddr(), SpuSetVoiceAR(), SpuSetVoiceDR(), SpuSetVoiceSR(),SpuSetVoiceRR(), SpuSetVoiceSL(), SpuSetVoiceARAttr(), SpuSetVoiceSRAttr(), SpuSetVoiceRRAttr(),SpuSetVoiceADSR(), SpuSetVoiceADSRAttr().

## SpuQuit

Terminates SPU processing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuQuit** (void)

### Arguments

None.

### Explanation

Terminates SPU processing. Normally, during a game, all devices, including SPU, are usually reset with a hardware reset, so it is not necessary to call SpuQuit(), but because SpuQuit is called with the original debug environment, the item below is reset in the current specification.

After this setting is made, DMA transfer to the sound buffer cannot be used

### Return value

None.

### Remarks

**See also:** SpuInit (p.), SpuInitHot (p.).

## SpuRead

Transfers data from the sound buffer to main memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuRead** (**addr*, *size*)
**unsigned char** **addr*;
**unsigned long** *size*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to transfer data start address in main memory |
| *size* | Transfered data size (in bytes) |

### Explanation

Transfers *size* bytes of data from the sound buffer to main memory *addr*.

The transfer destination main memory address *addr* must fulfill the following conditions.

- It is an address of an allocated variable that is a global variable
- It is an address of an allocated variable that is in the heap and is allocated by a function such as malloc.

That is, it does not address a stack area (a variable (= auto variable) declared in a function.

### Return value

Transferred data size.

If the specified data size is larger than 512 KB, the actual transferred size is returned.

### Remarks

**See also:** SpuWrite (p. 13-121), SpuSetTransferStartAddr (p. 13-87), SpuGetTransferStartAddr (p.).

## SpuReadDecodeData

Transfers sound data decoded by the SPU from the sound buffer to main memory.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuReadDecodeData** (**d_data*, *flag*)
**SpuDecodeData** **d_data;*
**unsigned long** *flag;*

### Arguments

| | | |
|---|---|---|
| *d_data* | Pointer to start address of SpuDecodeData structure in main memory | |
| *flag* | SPU_CDONLY | Set transfer of CD input only |
| | SPU_ALL | Set transfer of all data |

### Explanation

Transfers waveform data decoded by the SPU from the sound buffer to main memory.

The SPU writes sound data after CD input volume processing and sound data after Voice 1 and Voice 3 envelope processing to the sound buffer's starting 0x1000 byte (0x800 short int) area 16 bits (1 short int) at a time at each clock (44.1 kHz). Each piece of sound data has 0x400 byte (0x200 short int) buffers.

Data is signed 16-bit data, so access is in units of 16 bits (1 short int). Data is arranged as shown below.

**Table 13–10: Arrangement of Data**

| Map (short int) | Data Contents |
|---|---|
| 0x000 - 0x1ff | CD Left channel |
| 0x200 - 0x3ff | CD Right channel |
| 0x400 - 0x5ff | Voice 1 |
| 0x600 - 0x7ff | Voice 3 |

These are divided into the first half (0x100 short int) and the second half (0x100 short int); which buffer area is currently being written to is decided by the return value.

The main memory address addr storing the transfer data must fulfill the following conditions.

- It is an address of an allocated variable that is a global variable
- It is an address of an allocated variable that is in the heap and is allocated by a function such as malloc.

That is, it does not address a stack area (a variable (= auto variable) declared in a function.

### Return value

Returns the buffer area currently being written to, as shown below.

The return value is the area currently being written to, so data that can actually be used is in the area not being reported.

**Table 13–11**

| Return value | Meaning |
|---|---|
| SPU_DECODE_FIRSTHALF | Writes the first half of data |
| SPU_DECODE_SECONDHALF | Writes the second half of data |

### Remarks

**See also:** SpuWrite (p. 13-121), SpuSetTransferStartAddr (p.), SpuGetTransferStartAddr (p. 13-28), SpuDecodeData (p. 13-3).

## SpuReserveReverbWorkArea

Reserve/release reverb work area.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuReserveReverbWorkArea** (*on_off*)
**long** *on_off*;

### Arguments

| | |
|---|---|
| *on_off* | Reserve/release flag |

### Explanation

Reserves the current reverb work area corresponding to the current reverb mode in such a way that it is not allocated by SpuMalloc()/SpuMallocWithStartAddr(), or releases it so that it is allocated.

*on_off* specifies which action is performed. These settings are explained below.

**Table 13–12**

| Value | Description |
|---|---|
| SPU_ON | Reserve reverb work area |
| SPU_OFF | Release reverb work area |

a) SPU_ON

Reserves the reverb work area so that it is not an area allocated by SpuMalloc()/SpuMallocWithStartAddr(). Reserves the area without regard to reverb ON/OFF.

If the reverb work area has already been allocated by SpuMalloc() / SpuMallocWithStartAddr() as another area, it is not allocated and SPU_OFF is returned.

b) SPU_OFF

Releases the reverb work area so that it can be allocated by SpuMalloc() / SpuMallocWithStartAddr() as another area. Releases it regardless of reverb ON/OFF; reverb must have been turned off beforehand.

### Return value

When *on_off* is set to SPU_ON, if the reverb work area has already been allocated by SpuMalloc()/SpuMallocWithStartAddr() as another area, it is not reserved and SPU_OFF is returned. If it is reserved, SPU_ON is returned.

When *on_off* is set to SPU_OFF, SPU_OFF is always returned.

### Remarks

**See also:** SpuIsReverbWorkAreaReserved (p.), SpuSetReverbModeParam (p. 13-82), SpuSetReverb (p. 13-80), SpuMalloc (p. 13-55), SpuMallocWithStartAddr (p. 13-56).

## SpuRGetAllKeysStatus

Checks key on/key off for the specified range of voices.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.1* | *7/31/96* |

### Syntax

**void SpuRGetAllKeysStatus** (*min, max, *status*)
**long** *min*;
**long** *max*;
**char** **status*[24];

### Arguments

| | |
|---|---|
| *min* | Lower limit of the voice number to be checked |
| *max* | Upper limit of the voice number to be checked |
| *status[24]* | Pointer to the result of checking a voice |

### Explanation

Checks key on/key off and envelope status of all voices whose range is specified by *min* and *max*; checks actual key on/key off.

An error may result if multiple envelopes are set, and if volume goes to 0 in the course of changing envelope status.

The current key on/key off and envelope status of each voice is returned to status[24].

**Table 13–13**

| Value | Status |
|---|---|
| SPU_ON | Key on status |
| | Not turned off by SpuSetKey |
| | Envelope not 0 |
| SPU_ON_ENV_OFF | Key on status |
| | Not turned off by SpuSetKey |
| | Envelope 0 |
| SPU_OFF_ENV_ON | Key off status |
| | Turned off by SpuSetKey |
| | Envelope not 0 |
| SPU_OFF | Key off status |
| | Turned off by SpuSetKey |
| | Envelope 0 |

### Return value

| | |
|---|---|
| SPU_INVALID_ARGS | Invalid voice range |
| SPU_SUCCESS | Keys status contained in *status[24]*. |

### Remarks

**See also:** SpuSetKey (p.), SpuGetKeyStatus (p.).

## SpuRSetVoiceAttr

Sets attributes of each voice in the designated range.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.1* | *7/31/96* |

### Syntax

**long void SpuRSetVoiceAttr** (*min, max, *attr*)
**long** *min;*
**long** *max;*
**SpuVoiceAttr** **attr;*

### Arguments

| | |
|---|---|
| *min* | Lower limit of the voice number to be checked |
| *max* | Upper limit of the voice number to be checked |
| *attr* | Pointer to voice attributes |

### Explanation

Sets attributes for each voice, the range of which is specified by min and max.

Explicitly set voices by ORing together SPU_0CH, SPU_1CH, ...SPU_23CH in attr.voice, where the range of voices is specified by min and max.

You can set each attribute in attr.voice by ORing together the terms shown below.

**Table 13–14**

| Attribute | Description |
|---|---|
| SPU_VOICE_VOLL | Volume (left) |
| SPU_VOICE_VOLR | Volume (right) |
| SPU_VOICE_VOLMODEL | Volume mode (left) |
| SPU_VOICE_VOLMODER | Volume mode (right) |
| SPU_VOICE_PITCH | Interval (pitch specification) |
| SPU_VOICE_NOTE | Interval (note specification) |
| SPU_VOICE_SAMPLE_NOTE | Waveform data sample note |
| SPU_VOICE_WDSA | Waveform data start address |
| SPU_VOICE_ADSR_AMODE | ADSR Attack rate mode |
| SPU_VOICE_ADSR_SMODE | ADSR Sustain rate mode |
| SPU_VOICE_ADSR_RMODE | ADSR Release rate mode |
| SPU_VOICE_ADSR_AR | ADSR Attack rate |
| SPU_VOICE_ADSR_DR | ADSR Decay rate |
| SPU_VOICE_ADSR_SR | ADSR Sustain rate |
| SPU_VOICE_ADSR_RR | ADSR Release rate |
| SPU_VOICE_ADSR_SL | ADSR Sustain level |
| SPU_VOICE_ADSR_ADSR1 | ADSR adsr1 for 'VagAtr' |
| SPU_VOICE_ADSR_ADSR2 | ADSR adsr2 for 'VagAtr' |
| SPU_VOICE_LSAX | Loop start address |

If attr.mask is 0, set all attributes.

The individual settings of each attribute are described in SpuSetVoiceAttr.

## Return value

| | |
|---|---|
| SPU_INVALID_ARGS | Invalid voice range. |
| SPU_SUCCESS | Voice attributes set for specified range. |

## Remarks

**See also:** SpuGetVoiceAttr (p.), SpuSetKey (p.), SpuSetKeyOnWithAttr (p.).

## SpuSetCommonAttr

Sets attributes common to all voices (infrequent change requests).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuSetCommonAttr** (**attr*)
**SpuCommonAttr** **attr;*

### Arguments

*attr* Pointer to attributes common to all voices

### Explanation

Sets attributes common to all voices.

You can set each attribute (members of attr) in attr.mask by ORing together the terms shown below. If attr.mask is 0, all attributes are set.

**Table 13–15**

| Attribute | Description |
|---|---|
| SPU_COMMON_MVOLL | Master volume (left) |
| SPU_COMMON_MVOLR | Master volume (right) |
| SPU_COMMON_MVOLMODEL | Master volume mode (left) |
| SPU_COMMON_MVOLMODER | Master volume mode (right) |
| SPU_COMMON_CDVOLL | CD input volume (left) |
| SPU_COMMON_CDVOLR | CD input volume (right) |
| SPU_COMMON_CDREV | CD input reverb ON/OFF |
| SPU_COMMON_CDMIX | CD input ON/OFF |
| SPU_COMMON_EXTVOLL | External digital input volume (left) |
| SPU_COMMON_EXTVOLR | External digital input volume (right) |
| SPU_COMMON_EXTREV | External digital input reverb ON/OFF |
| SPU_COMMON_EXTMIX | External digital input ON/OFF |

Individual setting parameters are explained below.

a) Master Volume and Master Volume Mode

Master volume is set in attr.mvol; master volume mode is set in attr.mvolmode. Left and right are set independently.

The volume range obtainable and the various modes are the same as the settings for each voice; see Table 13-35 under SpuSetVoiceAttr().

b) CD Input Volume

CD input volume is set independently for left and right in attr.cd.volume in the range -0x8000 - 0x7fff. If the volume set is negative, the phase is inverted.

c) CD Input Reverb On/Off

Reverb is set in attr.cd.reverb. The values that may be specified are as follows.

**Table 13–16**

| Value | Description |
|---|---|
| SPU_ON | Set reverb on |
| SPU_OFF | Set reverb off |

d) CD Input Mixing On/Off

Sets CD input mixing in attr.cd.mix. The values that may be specified are as follows. CD input is not output unless this value is on.

**Table 13–17**

| Value | Description |
|---|---|
| SPU_ON | Set mixing on |
| SPU_OFF | Set mixing off |

e) External Digital Input Volume

External digital input volume is set independently for left and right in attr.ext.volume in the range -0x8000 - 0x7fff. If the volume set is negative, the phase is inverted.

f) External Digital Input Reverb On/Off

Reverb is set in attr.ext.reverb. The values that may be specified are as follows.

**Table 13–18**

| Value | Description |
|---|---|
| SPU_ON | Set reverb on |
| SPU_OFF | Set reverb off |

g) External Digital Input Mixing On/Off

Reverb is set in attr.cd.mix. The values that may be specified are as follows. External digital input is not output unless this value is on.

**Table 13–19**

| Value | Description |
|---|---|
| SPU_ON | Set mixing on |
| SPU_OFF | Set mixing off |

## Return value

None.

## Remarks

**See also:** SpuGetCommonAttr (p. 13-16), SpuSetVoiceAttr (p. 13-92).

## SpuSetEnv*

Sets basic sound library environment.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetEnv** *(*env)*

### Arguments

| | |
|---|---|
| *env* | Basic sound library environment attribute |

### Explanation

This function sets the basic sound library environment. Attribute can be set by setting env.mask with bitwise inclusive ORed desired attributes. Currently, there is only one available attribute; SPU_ENV_EVENT_QUEUEING (queue an event).

When env.mask is set to 0, all the attributes will be set.

See below for various setting;
- Queue an event
env.queueing,

SPU_ON ... Queue an event
SPU_OFF ... Do not queue an event (default) can set either to queue or not queue an event such as Key ON/OFF, Pitch LFO Voice Set, Noise Voice Set, and Reverb voice Set. Default is to set immediately without queuing.

### Return value

None.

### Remarks

**See also:** SpuSetKey(), SpuSetKeyOnWithAttr(), SpuSetPitchLFOVoice(), SpuSetNoiseVoice(), SpuSetReverbVoice(), SpuFlush().

## SpuSetIRQ

Sets interrupt request ON/OFF.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuSetIRQ** (*on_off*)
**long** *on_off;*

### Arguments

| | |
|---|---|
| *on_off* | Sets interrupt request ON/OFF/RESET |

**Table 13–20**

| Value | Description |
|---|---|
| SPU_ON | Set interrupt request |
| SPU_OFF | Cancel interrupt request |
| SPU_RESET | Reset interrupt request (= set after cancel) |

### Explanation

Sets interrupt request ON/OFF.

### Return value

Set value.

**Table 13–21**

| Value | Description |
|---|---|
| SPU_ON | Set interrupt request |
| SPU_OFF | Cancel interrupt request |
| SPU_RESET | Reset interrupt request |

### Remarks

**See also:** SpuGetIRQ (p.).

## SpuSetIRQAddr

Sets interrupt request address.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuSetIRQAddr** (*addr*)
**unsigned long** *addr;*

### Arguments

| | |
|---|---|
| *addr* | Interrupt request address |

### Explanation

Sets interrupt request address value. The address value must be

- In bytes
- Divisible by 8
- Less than 512 KB

### Return value

Returns the value of the address that is set.

If the value of the set address addr is not divisible by 8, the set value is advanced to the next value divisible by 8, and that value is set and returned.

If the address exceeds 512 KB, 0 is returned.

### Remarks

**See also:** SpuGetIRQAddr (p. 13-18), SpuSetIRQ (p.), SpuGetIRQ (p.).

## SpuSetIRQCallback

Sets callback at the time of an interrupt request.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**SpuSetIRQCallback** (*func*)
**SpuIRQCallbackProc** *func;*

### Arguments

| | |
|---|---|
| *func* | The callback function activated at the time of an interrupt request |

### Explanation

Sets a callback function activated at the time of an interrupt request.

If the callback function value is set to NULL, the callback is cleared.

### Return value

Pointer to the previously set function.

### Remarks

**See also:** SpuSetIRQ (p.), SpuSetIRQAddr (p.).

## SpuSetKey

Sets key on/key off for each voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuSetKey** (*on_off*, *voice_bit*)
**long** *on_off;*
**unsigned long** *voice_bit;*

### Arguments

| | |
|---|---|
| *on_off* | Sets key on/key off |
| *voice_bit* | Set voice |

### Explanation

Sets each voice specified by voice_bit as key on/key off.

Values that may be set by on_off are as follows.

**Table 13–22**

| Value | Description |
|---|---|
| SPU_ON | Set key on |
| SPU_OFF | Set key off |

Sets voice_bit by ORing together SPU_0CH, SPU_1CH...SPU_23CH.

### Return value

None.

### Remarks

When setting key on for voice 0 and voice 2, call SpuSetKey() as follows.

```
SpuSetKey (SPU_ON,  /* set key on */
           SPU_0CH | SPU_2CH);   /* 0 ch and 2 ch */
```

**See also:** SpuSetKeyOnWithAttr (p.), SpuSetVoiceAttr (p.).

## SpuSetKeyOnWithAttr

Sets key on with attributes for voice using attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuSetKeyOnWithAttr** (**attr*)
**SpuVoiceAttr** **attr;*

### Arguments

*attr* Pointer to voice attributes

### Explanation

Specifies attributes for each voice and sets key on.

Explicitly specify the voices to be produced by ORing together SPU_0CH, SPU_1CH...SPU_23CH in attr.voice.

You can each each attribute  in attr.voice by ORing together the terms shown below.

**Table 13–23**

| Attribute | Description |
|---|---|
| SPU_VOICE_VOLL | Volume (left) |
| SPU_VOICE_VOLR | Volume (right) |
| SPU_VOICE_VOLMODEL | Volume mode (left) |
| SPU_VOICE_VOLMODER | Volume mode (right) |
| SPU_VOICE_PITCH | Interval (pitch specification) |
| SPU_VOICE_NOTE | Interval (note specification) |
| SPU_VOICE_SAMPLE_NOTE | Waveform data sample note |
| SPU_VOICE_WDSA | Waveform data start address |
| SPU_VOICE_ADSR_AMODE | ADSR Attack rate mode |
| SPU_VOICE_ADSR_SMODE | ADSR Sustain rate mode |
| SPU_VOICE_ADSR_RMODE | ADSR Release rate mode |
| SPU_VOICE_ADSR_AR | ADSR Attack rate |
| SPU_VOICE_ADSR_DR | ADSR Decay rate |
| SPU_VOICE_ADSR_SR | ADSR Sustain rate |
| SPU_VOICE_ADSR_RR | ADSR Release rate |
| SPU_VOICE_ADSR_SL | ADSR Sustain level |
| SPU_VOICE_ADSR_ADSR1 | ADSR adsr1 for VagAtr |
| SPU_VOICE_ADSR_ADSR2 | ADSR adsr2 for VagAtr |
| SPU_VOICE_LSAX | Loop start address |

If attr.mask is 0, all attributes will be set.

The individual settings of each attribute are described in SpuSetVoiceAttr.

### Return value

None.

### Remarks

**See also:** SpuSetKey (p.), SpuSetVoiceAttr (p.), SpuGetVoiceAttr (p.), SpuVoiceAttr (p. 13-6).

## SpuSetMute

Sets sound muting ON/OFF.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuSetMute** (*on_off*)
**long** *on_off;*

### Arguments

| | |
|---|---|
| *on_off* | Mute ON/OFF |

### Explanation

Sets sound muting ON/OFF. on_off setting values are as follows.

**Table 13–24**

| Value | Description |
|---|---|
| SPU_ON | Mute on |
| SPU_OFF | Mute off |

However, CD input and external digital input are not muted by this mute ON/OFF.

### Return value

Set value.

| | |
|---|---|
| SPU_ON | Mute on |
| SPU_OFF | Mute off |

### Remarks

**See also:** SpuGetMute (p. 13-20).

## SpuSetNoiseClock

Sets noise source clock.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuSetNoiseClock** (*n_clock*)
**long** *n_clock;*

### Arguments

| | |
|---|---|
| *n_clock* | Noise source clock |

### Explanation

Set noise source clock value in n_clock. The clock value n_clock must be 0-0x3f.

### Return value

Noise source clock value set.

### Remarks

**See also:** SpuGetNoiseClock (p. 13-21), SpuSetNoiseVoice (p. 13-78), SpuGetNoiseVoice (p. 13-22).

## SpuSetNoiseVoice

Sets noise source ON/OFF for each voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuSetNoiseVoice** (*on_off*, *voice_bit*)
**long** *on_off*;
**unsigned long** *voice_bit*;

### Arguments

| | |
|---|---|
| *on_off* | Sets noise source ON/OFF |
| *voice_bit* | Set voice |

### Explanation

Sets each voice specified by *voice_bit* as noise on/noise off (i.e., use/do not use noise).

Values that may be set by *on_off* are as follows:

**Table 13–25**

| Value | Description |
|---|---|
| SPU_ON | Sets noise source |
| SPU_OFF | Releases noise source |

Specify the voices set in voice_bit by ORing together SPU_0CH-SPU_23CH.

### Return value

Returns the noise source ON/OFF value of the current voice. OR together SPU_0CH-SPU_23CH.

Distinguishes the noise source ON/OFF value by ANDing the return value and SPU_xxCH(xx=0~23).

**Table 13–26**

| Result of AND | Description |
|---|---|
| 0 | Sets noise source off |
| Other than 0 | Sets noise source on |

### Remarks

Set voice 0 and voice 2 noise source on as follows:

```
SpuSetNoiseVoice(SPU_ON,      /*set noise source on*/
   SPU_0CH | SPU_2CH);       /*0 ch and 2 ch*/
```

**See also:** SpuSetNoiseClock (p. 13-77), SpuGetNoiseClock (p.), SpuGetNoiseVoice (p.).

## SpuSetPitchLFOVoice

Sets pitch LFO ON/OFF for each voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuSetPitchLFOVoice** (*on_off*, *voice_bit*)
**long** *on_off*;
**unsigned long** *voice_bit;*

### Arguments

| | | |
|---|---|---|
| *on_off* | SPU_ON | Sets pitch LFO on |
| | SPU_OFF | Sets pitch LFO off |
| *voice_bit* | Sets voice | |

### Explanation

Sets pitch LFO ON/OFF for each voice.

Voice n, having pitch LFO set on, is set so that LFO sets pitch when the volume of voice (n-1) undergoes a time change. To make this pitch LFO valid, voice n and voice (n-1) must produce sound, and the volume of voice (n-1) must be set to 0 in advance. Voice (n - 1) can produce sound at an optional timing after voice n produces sound; LFO is applied at the moment when voice (n-1) produces sound.

Specify the voices set in voice_bit by ORing together SPU_0CH, SPU_1CH...SPU_23CH.

### Return value

Returns the pitch LFO ON/OFF value of the current voice. OR together SPU_0CH, SPU_1CH...SPU_23CH.

Distinguishes the pitch LFO ON/OFF value by ANDing the return value and SPU_xxCH (xx=0~23).

**Table 13–27**

| Result of AND | Description |
|---|---|
| 0 | Sets pitch LFO off |
| Other than 0 | Sets pitch LFO on |

### Remarks

**See also:** SpuGetPitchLFOVoice (p. 13-23), SpuSetKey (p.), SpuSetKeyOnWithAttr (p.).

## SpuSetReverb

Sets reverb ON/OFF.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuSetReverb** *(on_off)*
**long** *on_off;*

### Arguments

| | | |
|---|---|---|
| *on_off* | SPU_ON | Set reverb on |
| | SPU_OFF | Set reverb off |

### Explanation

Sets reverb ON/OFF.

If a reverb work area is not reserved with SpuReserveReverbWorkArea, when SPU_ON is specified by on_off, SpuSetReverb checks whether the area used as a work area by SpuMalloc/ SpuMallocWithStartAddr is being used as another area, and if it is being used, reverb is set off and SPU_OFF is returned.

If it is not being used, reverb is set on and SPU_ON is returned. When a reverb work area is reserved, an on_off value of SPU_ON sets reverb and returns SPU_ON.

### Return value

Set value

| | |
|---|---|
| SPU_ON | Reverb on |
| SPU_OFF | Reverb off |

### Remarks

**See also:** SpuGetReverb (p. 13-24), SpuSetReverbModeParam (p.), SpuReserveReverbWorkArea (p. 13-63).

## SpuSetReverbDepth

Sets the reverb depth parameter.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuSetReverbDepth** (**attr*)
**SpuReverbAttr** **attr;*

### Arguments

*attr* Pointer to reverb attribute

### Explanation

Sets the reverb depth parameter attribute.

You can set each attribute (members of attr) in attr.mask by ORing together the terms shown below.

**Table 13–28**

| Value | Description |
|---|---|
| SPU_REV_DEPTHL | Reverb depth (left) |
| SPU_REV_DEPTHR | Reverb depth (right) |

If attr.mask is 0, left and right attributes are set simultaneously.

a) Reverb Depth

Reverb depth is set independently for left and right. The range for this specification is -0x8000 -0x7fff.

If the value set is negative, the reverb sound (wet) phase is inverted.

### Return value

Always returns 0.

### Remarks

**See also:** SpuSetReverbModeParam (p.), SpuGetReverbModeParam (p.), SpuRevervbAttr (p. 13-5).

## SpuSetReverbModeParam

Sets reverb mode and parameters.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuSetReverbModeParam** (**attr*)
**SpuReverbAttr** **attr;*

### Arguments

*attr* Pointer to reverb attributes

### Explanation

Sets reverb mode and parameter attributes.

You can set each attribute (members of attr) in attr.mask by ORing together the terms shown below. If attr.mask is 0, all attributes are set.

**Table 13–29**

| Attribute | Description |
|---|---|
| SPU_REV_MODE | Mode setting |
| SPU_REV_DEPTHL | Reverb depth (left) |
| SPU_REV_DEPTHR | Reverb depth (right) |
| SPU_REV_DELAYTIME | Delay time (ECHO, DELAY only) |
| SPU_REV_FEEDBACK | Feedback (ECHO, DELAY only) |

a) Reverb Mode (Table 8-31)
Sets reverb mode. Setting attributes other than "Depth" (reverb depth) varies according to the reverb mode.

**Table 13–30: Reverb Mode and Other Attributes**

| attr.mode | mode | Delay time | Feedback |
|---|---|---|---|
| SPU_REV_MODE_OFF | off | | |
| SPU_REV_MODE_ROOM | room | | |
| SPU_REV_MODE_STUDIO_A | | studio (small) | |
| SPU_REV_MODE_STUDIO_B | | studio (med) | |
| SPU_REV_MODE_STUDIO_C | | studio (big) | |
| SPU_REV_MODE_HALL | hall | | |
| SPU_REV_MODE_SPACE | space echo | | |
| SPU_REV_MODE_ECHO | echo | can set | can set |
| SPU_REV_MODE_DELAY | delay | can set | can set |
| SPU_REV_MODE_PIPE | half echo | | |

When reverb mode is changed (this happens even at initial setting because the initial value is SPU_REV_MODE_OFF), the internal reverb Depth value is 0 even if Depth was previously set in SpuSetReverbModeParam(). This is because the work area size changes when this mode changes, so incorrect data in the work area produces noise. So after the reverb mode changes, Depth needs to be reset in SpuSetReverbModeParam() or SpuSetReverbDepth().

Based on reverb characteristics, the time to complete one scan of the work area is estimated and the mode/depth are set; or, after the mode is set, the work area data is erased then Depth is set (to be described later).

The sound buffer volume occupied by the work area depends on the reverb mode as shown in Table 8-31. However, this area is managed by a memory management mechanism such as SpuMalloc(). See SpuMalloc() for details.

**Table 13–31: Volume Occupied by Reverb Mode In Sound Buffer**

| attr.mode | mode | hexadecimal | decimal |
|---|---|---|---|
| SPU_REV_MODE_OFF | off | 0/80 (*) | 0/128 (*) |
| SPU_REV_MODE_ROOM | room | 26c0 | 9920 |
| SPU_REV_MODE_STUDIO_A | | studio (small) | 1f40 8000 |
| SPU_REV_MODE_STUDIO_B | | studio (med) | 4840 18496 |
| SPU_REV_MODE_STUDIO_C | | studio (big) | 6fe0 28640 |
| SPU_REV_MODE_HALL | hall | ade0 | 44512 |
| SPU_REV_MODE_SPACE | space echo | f6c0 | 63168 |
| SPU_REV_MODE_ECHO | echo | 18040 | 98368 |
| SPU_REV_MODE_DELAY | delay | 18040 | 98368 |
| SPU_REV_MODE_PIPE | half echo | 3c00 | 15360 |

(*) If SpuReserveReverbWorkArea (SPU_ON) is used for address setting, it takes 128 bytes even if the mode is off. If SpuReserveReverbWorkArea (SPU_OFF) is used, it takes 0 bytes.

If SPU_REV_MODE_CLEAR_WA is ORed in attr.mode, it clears the area needed by reverb mode set when setting reverb mode. This is a measure against noise when changing modes. However, the sound buffer is cleared by synchronous DMA transfer, so other processing (drawing, sound generation) is not performed during this processing, and some wait time is needed, depending on the reverb type.

SpuClearReverbWorkArea() is used to forcibly clear the area used by the reverb mode specified when setting reverb mode with optional timing.

b) Reverb Depth

Set in attr.depth, independently for left and right. Values are set in the range -0x8000 - 0x7fff.

If the set value is negative, the reverb sound (wet) phase is inverted.

c) Delay Time

Set in attr.delay. Values are set in the range 0-127. Valid when mode is SPU_REV_MODE_ECHO or SPU_REV_MODE_DELAY.

d) Feedback

Valid when mode is SPU_REV_MODE_ECHO or SPU_REV_MODE_DELAY.

Delay time is set in attr.feedback with values from 0 to 127.

## Return value

If the area used as a work area by the new mode is being used as another area by SpuMalloc()/SpuMallocWithStartAddr(), none of the set reverb attributes are set and SPU_ERROR is returned. If it is not being used, the set reverb attributes are set and 0 is returned.

SPU_ERROR is also returned when an invalid SPU_REV_MODE is set.

## Remarks

**See also:** SpuGetReverbModeParam (p.), SpuMalloc (p. 13-55), SpuMallocWithStartAddr (p. 13-56), SpuReserveReverbWorkArea (p. 13-63), SpuClearReverbWorkArea (p. 13-12).

## SpuSetReverbVoice

Sets reverb ON/OFF for each voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuSetReverbVoice** (*on_off*, *voice_bit*)
**long** *on_off*;
**unsigned long** *voice_bit*

### Arguments

| | |
|---|---|
| *on_off* | Sets reverb ON/OFF |
| *voice_bit* | Set voice |

### Explanation

Sets each voice specified by *voice_bit* as reverb on/reverb off.

Values that may be set by *on_off* are as follows:

**Table 13–32**

| Value | Description |
|---|---|
| SPU_ON | Set reverb on |
| SPU_OFF | Set reverb off |

Specify the voices set in *voice_bit* by ORing together SPU_0CH-SPU_23CH.

### Return value

Returns the reverb ON/OFF value of the current voice. OR together SPU_0CH-SPU_23CH.

Distinguishes the noise source ON/OFF value by ANDing the return value and SPU_xxCH(xx=0~23).

**Table 13–33**

| Result of AND | Description |
|---|---|
| 0 | Sets reverb off |
| Other than 0 | Sets reverb on |

### Remarks

Set voice 0 and voice 2 reverb on as follows:

```
SpuSetReverbVoice(SPU_ON,  /*set reverb on*/
   SPU_0CH | SPU_2CH);     /*0 ch and 2 ch*/
```

**See also:** SpuGetReverbVoice (p.).

## SpuSetTransferCallback

Sets callback function when DMA transfer is completed.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.1* | *7/31/96* |

### Structure

**SpuTransferCallbackProc SpuSetTransterCallback** (SpuTransferCallbackProc *func*)

### Arguments

| | |
|---|---|
| *func* | Starts callback function when DMA transfer is completed. |

### Explanation

Sets callback function started when DMA transfer is completed.

If the value of the callback function is set to NULL, the callback is cleared.

When a callback is set using SpuSetTransferCallback() and starting at DMA transfer completion, SpuIsTransferCompleted does not function.

### Return value

This functions returns the previously set callback function. If a callback function is not set, the function returns NULL.

### Remarks

**See also:** SpuWrite (p. 13-121), SpuWrite0 (p.), SpuWritePartly (p.), SpuRead ( p. 13-60), SpuIsTransferCompleted (p.).

## SpuSetTransferMode

Sets sound buffer transfer mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**long SpuSetTransferMode** *(mode)*
**long** *mode;*

### Arguments

| | | |
|---|---|---|
| *mode* | SPU_TRANSFER_BY_DMA | DMA transfer setting |
| | SPU_TRANSFER_BY_IO | I/O transfer setting |

### Explanation

Sets mode when transferring data from main memory to the sound buffer.

Mode values are as shown below. DMA transfer is the default.

**Table 13–34**

| Value | Description |
|---|---|
| SPU_TRANSFER_BY_DMA | DMA transfer setting<br>Can do other processing during transfer |
| SPU_TRANSFER_BY_IO | I/O transfer setting<br>Transfer uses the CPU;<br>cannot do other processing during transfer |

These specifications are valid only when transferring data from main memory to the sound buffer. DMA transfer is always used when transferring data from the sound buffer to main memory.

When transfer is done without first calling this function, transfer mode is set to a previously determined value.

### Return value

Set transfer mode

| | |
|---|---|
| SPU_TRANSFER_BY_DMA | DMA transfer setting |
| SPU_TRANSFER_BY_IO | I/O transfer setting |

### Remarks

**See also:** SpuGetTransferMode (p.), SpuWrite (p. 13-121).

## SpuSetTransferStartAddr

Sets sound buffer transfer destination/transfer source start address.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuSetTransferStartAddr** (*addr*)
**unsigned long** *addr;*

### Arguments

| | |
|---|---|
| *addr* | Sound buffer transfer destination/transfer source start address |

### Explanation

Sets a starting address specified in addr for transferring from main memory to the sound buffer, and from the sound buffer to main memory.

However, the start address value must be

- In bytes.
- Divisible by 8.
- Greater than 0x100f and less than 512 KB for transfers to the sound buffer.
- Between 0x0-0xfff for transfers from the sound buffer.

For transfers from the 0x0 - 0xfff area, see SpuReadDecodeData(). 0x1000 - 0x100f is reserved for the system.

### Return value

Set start address value.

If the value of the set address addr is not divisible by 8, the set value is advanced to the next value divisible by 8 and that value is returned.

For values smaller than 0x100f or greater than 512 KB, 0 is returned.

### Remarks

**See also:** SpuGetTransferStartAddr (p. 13-28), SpuWrite (p. 13-121), SpuRead (p. 13-60).

## SpuSetVoiceADSR*

Sets ADSR.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceAR** *(***int** *voiceNum,* **unsigned short** **AR,* **unsigned short** **DR*
**unsigned short** **SR,* **unsigned short** **RR,*
**unsigned short** **SL)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |
| *DR* | ADSR decay rate |
| *SR* | ADSR sustain rate |
| *RR* | ADSR release rate |
| *SL* | ADSR sustain level |

### Explanation

This function sets each ADSR attribute used in the S voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_AR
SPU_VOICE_ADSR_DR
SPU_VOICE_ADSR_SR
SPU_VOICE_ADSR_RR
SPU_VOICE_ADSR_SL

For attack, sustain, and  release rate, rate mode becomes as below:

--------------------+----------------------------------

Attack Rate Mode | SPU_VOICE_LINEARIncN (Linear Increase)

Sustain Rate Mode | SPU_VOICE_LINEARDecN (Linear Decrease)

Release_Rate_Mode | SPU_VOICE_LINEARDecN_(Linear Decrease)

--------------------+----------------------------------

If you want to set multiple rate modes at the same time, use SpuSetVoiceADSRAttr.

Refer to SpuSetVoiceAttr for values that can be specified in each rate.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceAR(), SpuSetVoiceDR(), SpuSetVoiceSR(), SpuSetVoiceRR(), SpuSetVoiceSL().

## SpuSetVoiceADSRAttr*

Sets ADSR and each mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceARAttr** *(***int** *voiceNum,* **unsigned short** **AR,* **unsigned short** **DR*
**unsigned short** **SR,* **unsigned short** **RR,* **unsigned short** **SL*
**long** **ARmode,* **long** **SRmode,* **long** **RRmode)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |
| *DR* | ADSR decay rate |
| *SR* | ADSR sustain rate |
| *RR* | ADSR release rate |
| *SL* | ADSR sustain level |
| *ARmode* | ADSR attack rate mode |
| *SRmode* | ADSR sustain rate mode |
| *RRmode* | ADSR release rate mode |

### Explanaton

This function sets ADSR attributes and mode, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_AR, SPU_VOICE_ADSR_AMODE
SPU_VOICE_ADSR_DR
SPU_VOICE_ADSR_SR, SPU_VOICE_ADSR_SMODE
SPU_VOICE_ADSR_RR, SPU_VOICE_ADSR_RMODE
SPU_VOICE_ADSR_SL

Refer to SpuSetVoiceAttr for values that can be specified in each rate and rate mode.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceADSR(), SpuSetVoiceAR(), SpuSetVoiceDR(), SpuSetVoiceSR(), SpuSetVoiceRR(), SpuSetVoiceSL(), SpuSetVoiceARAttr(), SpuSetVoiceSRAttr(), SpuSetVoiceRRAttr().

## SpuSetVoiceAR*

Sets ADSR attack rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceAR** *(***int** *voiceNum,* **unsigned short** **AR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |

### Explanation

This function sets ADSR attack rate in voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_AR

ADSR attack rate mode becomes SPU_VOICE_LINEARIncN (Linear increase mode) . If you want to set ADSR attack rate and ADSR attack rate mode at the same time, use SpuSetVoiceARAttr.

Refer to SpuSetVoiceAttr for values that can be specified in ADSR attack rate, AR.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceARAttr().

## SpuSetVoiceARAttr*

Sets ADSR attack rate / attack rate mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceARAttr** *(***int** *voiceNum,* **unsigned short** **AR,* **long** **ARmode)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *AR* | ADSR attack rate |
| *Armode`* | ADSR attack rate mode |

### Explanation

This function sets ADSR attack rate / ADSR attack rate mode used in voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_AR
SPU_VOICE_ADSR_AMODE

Refer SpuSetVoiceAttr for values that can be specified in ADSR attack rate AR and ADSR attack rate mode, ARmode.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceAR().

## SpuSetVoiceAttr

Sets attributes for each voice.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuSetVoiceAttr** (**attr*)
**SpuVoiceAttr** **attr;*

### Arguments

*attr* Pointer to voice attributes

### Explanation

Sets attributes for each voice

Explicitly set the voices you wish to set by ORing together SPU_0CH, SPU_1CH, ...SPU_23CH in attr.voice.

You can set each attribute in attr.voice by ORing together the terms shown below.

**Table 13–35**

| Attribute | Description |
|---|---|
| SPU_VOICE_VOLL | Volume (left) |
| SPU_VOICE_VOLR | Volume (right) |
| SPU_VOICE_VOLMODEL | Volume mode (left) |
| SPU_VOICE_VOLMODER | Volume mode (right) |
| SPU_VOICE_PITCH | Interval (pitch specification) |
| SPU_VOICE_NOTE | Interval (note specification) |
| SPU_VOICE_SAMPLE_NOTE | Waveform data sample note |
| SPU_VOICE_WDSA | Waveform data start address |
| SPU_VOICE_ADSR_AMODE | ADSR Attack rate mode |
| SPU_VOICE_ADSR_SMODE | ADSR Sustain rate mode |
| SPU_VOICE_ADSR_RMODE | ADSR Release rate mode |
| SPU_VOICE_ADSR_AR | ADSR Attack rate |
| SPU_VOICE_ADSR_DR | ADSR Decay rate |
| SPU_VOICE_ADSR_SR | ADSR Sustain rate |
| SPU_VOICE_ADSR_RR | ADSR Release rate |
| SPU_VOICE_ADSR_SL | ADSR Sustain level |
| SPU_VOICE_ADSR_ADSR1 | ADSR adsr1 for 'VagAtr' |
| SPU_VOICE_ADSR_ADSR2 | ADSR adsr2 for 'VagAtr' |
| SPU_VOICE_LSAX | Loop start address |

If attr.mask is 0, all attributes will be set.

The individual settings are described below.

a) Volume and Volume Mode

Each Volume Mode and the range of possible volume settings for each Volume Mode are provided below.

**Table 13–36: Volume Mode and Volume Setting Ranges**

| Mode (phase) | SPU_VOICE_VOLMODEx | SPU_VOICE_VOLx |
|---|---|---|

| | | |
|---|---|---|
| Direct mode | SPU_VOICE_DIRECT | -0x4000 - 0x3fff |
| Linear inc. mode | SPU_VOICE_LINEARIncN | 0x00 - 0x7f (normal) |
| Linear inc. mode | SPU_VOICE_LINEARIncR | 0x00 - 0x7f (inverted) |
| Linear dec. mode | SPU_VOICE_LINEARDecN | 0x00 - 0x7f (normal) |
| Linear dec. mode | SPU_VOICE_LINEARDecR | 0x00 - 0x7f (inverted) |
| Expon. inc. mode | SPU_VOICE_EXPIncN | 0x00 - 0x7f (normal) |
| Expon. inc. mode | SPU_VOICE_EXPIncR | 0x00 - 0x7f (inverted) |
| Expon. dec. mode | SPU_VOICE_EXPDec | 0x00 - 0x7f |

- Direct Mode
  Fixed volume mode. In normal usage, this mode produces sound.
  When the set volume is negative, its phase is reversed. In this situation, "inverted phase", described below, is valid.
- Linear Increase Mode (Normal Phase)
  When the current volume value is positive and this mode is specified as the sound production status, volume increases linearly from the current value to the maximum value.
- Linear Increase Mode (Inverted Phase)
  When the current volume value is negative (inverted phase) and this mode is specified as the sound production status, volume increases linearly from the current value to the maximum value, with phase inverted.
- Linear Decrease Mode (Normal Phase)
  When the current volume value is positive and this mode is specified as the sound production status, volume decreases linearly from the current value to the minimum volume value.
- Linear Decrease Mode (Inverted Phase)
  When the current volume value is negative (inverted phase) and this mode is specified as the sound production status, volume decreases linearly from the current value to the minimum volume value, with phase inverted.
- Exponential Increase Mode (Normal Phase)
  When the current volume value is positive and this mode is specified as the sound production status, volume increases exponentially from the current value to the maximum value.
- Exponential Increase Mode (Inverted Phase)
  When the current volume value is negative (inverted phase) and this mode is specified as the sound production status, volume increases exponentially from the current value to the maximum value, with phase inverted.
- Exponential Decrease Mode
  When this mode is specified as the sound production status, whether the current volume value is positive or negative, volume decreases exponentially from the current value to the minimum volume value.

b) Interval (set pitch, set note)

Interval may be set by the two methods listed below.

- Pitch specification
  Specify an interval in attr.pitch in the range 0x0000-0x3fff.
  See Table 8-38 for an explanation of the meaning of these values. The only unit shown in the table is octaves, but any value in the range 0x0000-0x3fff may be set.

**Table 13–37: Pitch Specification Values and Interval**

| Value Set | 0x0200 | 0x0400 | 0x0800 | 0x1000 | 0x2000 | 0x3fff |
|---|---|---|---|---|---|---|
| Interval | - 3 oct. | - 2 oct. | - 1 oct. | tone | + 1 oct. | + 2 oct. |

- Note specification
  An interval is set in attr.note as follows, using a 16-bit value for note and cent (here, the value of a half tone divided by 128).

This setting cannot be used unless the waveform data sample note feature, described below, is set.

**Table 13–38: Note Specification Values**

| Bit | Value Set |
|---|---|
| Upper 8 bits | MIDI note number |
| Lower 8 bits | Cent (expressed as a half tone divided by 128) |

c) Waveform Data Sample Note

Sets interval in attr.sample_note at the time of sampling, using a 16-bit value for note and cent (here, the value of a half tone divided by 128). Setting this value makes it possible to set b) Interval--Note specification as above.

**Table 13–39: Waveform Data Sample Note Specification Values**

| Bit | Value Set |
|---|---|
| Upper 8 bits | MIDI note number |
| Lower 8 bits | Cent (expressed as half tone divided by 128) |

d) Waveform Data Start Address

The sound buffer starting address of the waveform data you want to produce in the voice is set in attr.addr.

e) Loop Start Address

If waveform data that generates sound in a voice is created with a loop specified, and if the waveform starting address is set, the loop start address is usually automatically identified and set. Explicit setting is unnecessary.

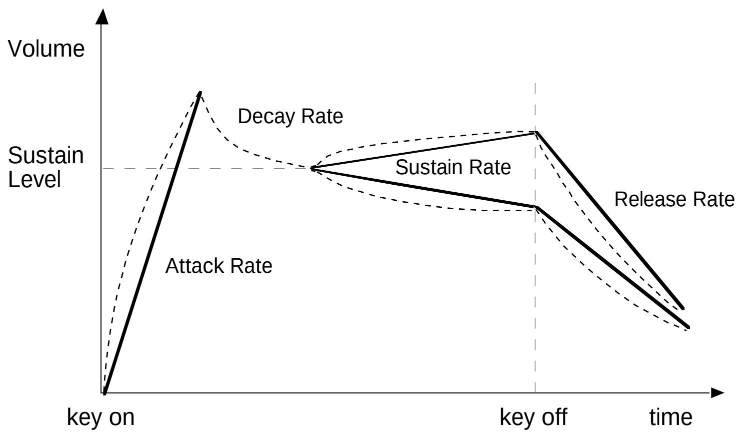
However, when you wish to set a loop start address dynamically at the time of execution, you must set the address that is the starting point of the loop in the sound buffer in attr.loop_addr.

If a loop was not set at the time of waveform data creation, even if SPU_VOICE_LSAX is specified and set in attr.loop_addr, that setting is invalid.

f) ADSR

A conceptual diagram of ADSR is shown below.

**Figure 13–1: ADSR Conceptual Diagram**



ADSR attributes are set by the structure members listed in Table 8-41; the range of these attributes is listed in Table 8-41.

**Table 13–40: Parameters and Structure Members**

| | Attribute | Structure Member |
|---|---|---|
| Rate | Attack rate | attr.ar, attr.a_mode |
| | Decay rate | attr.dr |

| | | |
|---|---|---|
| | Sustain rate | attr.sr, attr.s_mode |
| | Release rate | attr.rr, attr.r_mode |
| Level | Sustain level | attr.sl |

**Table 13–41: Rate and Level Setting Ranges**

| Attribute | Structure Member | Setting Range |
|---|---|---|
| Attack rate | attr.ar | 0x00 - 0x7f |
| Decay rate | attr.dr | 0x0 - 0xf |
| Sustain rate | attr.sr | 0x00 - 0x7f |
| Release rate | attr.rr | 0x00 - 0x1f |
| Sustain level | attr.sl | 0x0 - 0xf |

Rate curves may be set for Attack, Sustain, Release (see Table 8-43).

Because only exponential decrease may be used for Decay, that attribute cannot be set.

**Table 13–42: ADSR Rate Modes**

| Attribute | Mode settable in attr.?_mode |
|---|---|
| Attack rate | SPU_VOICE_LINEARIncN (linear increase ) |
| | SPU_VOICE_EXPIncN (exponential increase) |
| Decay rate | N/A |
| Sustain rate | SPU_VOICE_LINEARIncN (linear increase ) |
| | SPU_VOICE_LINEARDecN (linear decrease) |
| | SPU_VOICE_EXPIncN (exponential increase) |
| | SPU_VOICE_EXPDec (exponential decrease) |
| Release rate | SPU_VOICE_LINEARDecN (linear decrease) |
| | SPU_VOICE_EXPDec (exponential decrease) |

Also, data from structure VagAtr members adsr1 and adsr2 may be set directly in attr.adsr1 and attr.adsr2. In this case only SPU_VOICE_ADSR_ADSR1 and SPU_VOICE_ADSR_ADSR2 can be set for ADSR in attr.mask.

## Return value

None.

## Remarks

**See also:** SpuRSetVoiceAttr (), SpuGetVoiceAttr (), SpuSetKey (), SpuSetKeyOnWithAttr (), SpuVoiceAttr (p. 13-6).

## SpuSetVoiceDR*

Sets ADSR decay rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceDR** *(***int** *voiceNum,* **unsigned short** **DR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *DR* | ADSR decay rate |

### Explanation

This function sets ADSR decay rate used in the voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_DR

Refer to SpuSetVoiceAttr for values that can be specified in ADSR decay rate, DR.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr().

## SpuSetVoiceLoopStartAddr*

Sets loop start address of waveform data in the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceLoopStartAddr** *(***int** *voiceNum,* **unsigned long** **loopStartAddr)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *loopStartAddr* | Loop start address |

### Explanation

This function sets start address of waveform data in the sound buffer, equivalent process to SpuSetVoiceAttr

SPU_VOICE_LSAX

Refer to SpuSetVoiceAttr for values that can be specified in the loopStartAddr, loop start address.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetTransferStartAddr().

## SpuSetVoiceNote*

Sets interval (note specification).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceNote** *(***int** *voiceNum,* **unsigned short** **note)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *note* | Interval (note specification) |

### Explanation

This function sets the voice interval by note, equivalent process to SpuSetVoiceAttr

SPU_VOICE_NOTE

Thus prior to call SpuSetVoiceNote, SpuSetVoiceAttr

SPU_VOICE_SAMPLE_NOTE

or the waveform data sample note feature for voice must be set. Refer to SpuSetVoiceAttr for values that can be specified in the interval by note specification.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceSampleNote().

## SpuSetVoicePitch*

Sets interval (pitch specification).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoicePitch** *(***int** *voiceNum,* **unsigned short** **pitch)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *pitch* | Interval (pitch specification) |

### Explanation

This function sets the voice interval by pitch, equivalent process to SpuSetVoiceAttr

SPU_VOICE_PITCH

Refer to SpuSetVoiceAttr for values that can be specified in the interval by pitch specification..

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr().

## SpuSetVoiceRR*

Sets ADSR release rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceRR** *(***int** *voiceNum,* **unsigned short** **RR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *RR* | ADSR release rate |

### Explanation

This function sets ADSR release rate used in the voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_RR

ADSR sustain rate mode becomes SPU_VOICE_LINEARDecN (Linear decrease mode). If you want to set ADSR release rate and ADSR release rate mode at the same time, use SpuSetVoiceRRAttr.

Refer to SpuSetVoiceAttr for values that can be specified in ADSR release rate, RR.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceRRAttr().

## SpuSetVoiceRRAttr*

Sets ADSR release rate / release rate mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceRRAttr** *(***int** *voiceNum,* **unsigned short** **RR,* **long** *RRmode*)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *RR* | ADSR release rate |
| *RRmode* | ADSR release rate mode |

### Explanation

This function sets ADSR release rate / ADSR release rate mode used in the voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_RR
SPU_VOICE_ADSR_RRMODE

Refer to SpuSetVoiceAttr for values that can be specified in ADSR release rate, RR and ADSR release rate mode, RRmode.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceRR().

## SpuSetVoiceSampleNote*

Sets waveform data sample note.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceSampleNote** *(***int** *voiceNum,* **unsigned short** **sampleNote)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *sampleNote* | Sets waveform data sample note |

### Explanation

This function sets the waveform data sample note for voice, equivalent process to SpuSetVoiceAttr

SPU_VOICE_SAMPLE_NOTE

Refer to SpuSetVoiceAttr for values that can be specified in sampleNote.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceNote().

## SpuSetVoiceSL*

Sets ADSR sustain level.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceSL** *(***int** *voiceNum,* **unsigned short** **SL)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *SL* | ADSR sustain level |

### Explanation

This function sets ADSR release rate used in the voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_SL

ADSR sustain level mode becomes SPU_VOICE_LINEARDecN (Linear decrease mode). If you want to set ADSR sustain level and ADSR sustain level mode at the same time, use SpuSetVoiceSLAttr.

Refer to SpuSetVoiceAttr for values that can be specified in ADSR sustain level, SL.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceRRAttr().

## SpuSetVoiceSR*

Sets ADSR sustain rate.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceSR** *(***int** *voiceNum,* **unsigned short** **SR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *SR* | ADSR sustain rate |

### Explanation

This function sets ADSR sustain rate used in the voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_SR

ADSR sustain rate mode becomes SPU_VOICE_LINEARDecN (Linear decrease mode). If you want to set ADSR sustain rate and ADSR sustain rate mode at the same time, use SpuSetVoiceSRAttr.

Refer to SpuSetVoiceAttr for values that can be specified in ADSR sustain rate, SR.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceSRAttr().

## SpuSetVoiceSRAttr*

Sets ADSR sustain rate / sustain rate mode.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceSRAttr** *(***int** *voiceNum,* **unsigned short** **SR,* **long** *SRmode*)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *SR* | ADSR sustain rate |
| *SRmode* | ADSR sustain rate mode |

### Explanation

This function sets ADSR sustain rate / ADSR sustain rate mode used in the voice, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_SR
SPU_VOICE_ADSR_SRMODE

Refer to SpuSetVoiceAttr for values that can be specified in ADSR sustain rate, SR and ADSR sustain rate mode, SRmode.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceSR().

## SpuSetVoiceStartAddr*

Sets start address of waveform data in the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceStartAddr** *(***int** *voiceNum,* **unsigned long** **startAddr)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice number (0 - 23) |
| *startAddr* | Waveform data start address |

### Explanation

This function sets start address of waveform data in the sound buffer, equivalent process to SpuSetVoiceAttr

SPU_VOICE_WDSA

Refer to SpuSetTransferStartAddr for values that can be specified in the startAddr, waveform data start address.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetTransferStartAddr().

## SpuSetVoiceVolume*

Voice volume set.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceVolume** *(***int** *voiceNum,* **short** **volumeL,* **short** **volumeR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice Number (0 - 23) |
| *volumeL* | Volume (Left) |
| *volumeR* | Volume (Right) |

### Explanation

This function sets the voice volume, equivalent process to SpuSetVoiceAttr

SPU_VOICE_VOLL
SPU_VOICE_VOLR

Thus the Volume Mode will become "Direct Mode" and the range of value that can be specified to volumeL and volumeR is equivalent to "Direct Mode" of SpuSetVoiceAttr. If you want to specify both volume and volume mode at the same time, use SpuSetVoiceVolumeAttr. Refer to SpuSetVoiceAttr for values that can be specified in volumeL and/or volumeR.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceVolumeAttr().

## SpuSetVoiceVolumeAttr*

Voice volume/volume mode set.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.6* | *10/23/96* |

### Syntax

**void SpuSetVoiceVolumeAttr** *(***int** *voiceNum,* **short** **volumeL,* **short** ***volumeR,* **short** **volModeL,* **short** **volModeR)*

### Arguments

| | |
|---|---|
| *voiceNum* | Voice Number (0 - 23) |
| *volumeL* | Volume (Left) |
| *volumeR* | Volume (Right) |
| *volModeL* | Volume mode (Left) |
| *volModeR* | Volume mode (Right) |

### Explanation

This function sets voice volume and/or volume mode, equivalent process to SpuSetVoiceAttr

SPU_VOICE_VOLL
SPU_VOICE_VOLR
SPU_VOICE_VOLMODEL
SPU_VOICE_VOLMODER

Refer to SpuSetVoiceAttr for values that can be specified in volModeL, volModeR, volumeL and/or volumeR.

### Return value

None.

### Remarks

**See also:** SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceVolumeAttr()

## SpuStart

Starts SPU processing.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**void SpuStart** (void)

### Arguments

None.

### Explanation

SpuStart() starts SPU processing. This function is also called by SpuInit(), so it is not necessary to call it when initializing, but SpuStart() must be called after calling SpuQuit() if you use SpuQuit() to turn functionality off.

In the current specification, DMA transfer initialization setting is performed after SpuStart() is called.

### Return value

None.

### Remarks

**See also:** SpuQuit (p.), SpuInit (p.).

## SpuStEnv*

SPU streaming environment attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *2.x* | *7/31/96* |

### Structure

```
typedef struct {
    long size;
    SpuStVoiceAttr voice[24];
} SpuStEnv
```

### Members

| | |
|---|---|
| *size* | Stream buffer size |
| *voice* | Each stream attribute set |

### Explanation

Used in SPU streaming library, streaming environment and each stream attribute setting.

### Remarks

**See also:** SpuStVoiceAttr (p. 13-8), SpuStInit (p. 13-113).

## SpuStGetStatus

Determines the SPU streaming state.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**long SpuStGetStatus (void)**

### Arguments

None.

### Explanation

It determines the state of the SPU streaming.

### Return value

**Table 13–43**

| Attribute | Description |
|---|---|
| SPU_ST_NOT_AVAILABLE | SPU streaming is not available; SpuStInit() has not been called. |
| SPU_ST_IDLE | Data transfer to the sound buffer has not been performed yet or all streams have terminated already. |
| SPU_ST_PREPARE | Transferring the first 1 buffer. |
| SPU_ST_TRANSFER | Transferring the data to the sound buffer.<br>If SpuStTransfer (SPU_ST_PREPARE, ) is executed in this state, the status does not change to SPU_ST_PREPARE. |
| SPU_ST_FINAL | Waiting for the end of the playback after transferring the last 1 buffer. SpuStTransfer() is not accepted in this state. |

### Remarks

**See also:** SpuStInit (p. 13-113), SpuStTransfer (p.), SpuStGetVoiceStatus (p. 13-112).

## SpuStGetVoiceStatus

Determines the voices used for SPU streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**unsigned long SpuStGetVoiceStatus (void)**

### Arguments

None.

### Explanation

It determines the voices used for the SPU streaming.

### Return Value

The value of the voices represented by the bitOR of SPU_0CH to SPU_23CH.

### Remarks

**See also:** SpuStTransfer (p.), SpuStGetStatus (p.).

## SpuStInit

Initializes SPU streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**SpuStEnv *SpuStInit** (*long mode*)

### Arguments

| | |
|---|---|
| *mode* | Not used under the current specification. Set "0". |

### Explanation

Initializes the streaming. *mode* is called only once in the executed program. SPU streaming is available after initialization.

### Return Value

Pointer to the SPU streaming environment structure SpuStEnv.

### Remarks

**See also:** SpuStQuit (p.), SpuStEnv (p. 13-7).

## SpuStQuit

Completes SPU streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**long SpuStQuit (void)**

### Arguments

None.

### Explanation

Completes the SPU streaming. Prior to calling this function, the termination processing must be completed for all the streams.

### Return value

| | |
|---|---|
| SPU_ST_ACCEPT | Normal end |
| SPU_ST_WRONG_STATUS | SpuStQuit is not accepted. The cause is the current status is not SPU_ST_IDLE. |

### Remarks

**See also:** SpuStInit (p.), SpuStGetStatus (p.).

## SpuStSetPreparationFinishedCallback

Sets the callback function called at the completion of the data transfer in the preparation for the stream in the SPU streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**SpuStCallbackProc SpuStSetPreparationFinshedCallback (***SpuStCallbackProc callback_proc***);**
**SpuStCallbackProc callback_proc (***unsigned long voice_bit, long status***)**

### Arguments

| | |
|---|---|
| *callback_proc* | Pointer towards the callback function called at the completion of the data transfer in the preparation for the stream. |

### Explanation

Sets the callback function called at the completion of the data transfer in the preparation for the stream in the SPU streaming.

When callback_proc is called, the value of the voices assigned for the stream where the data transfer is completed in the preparation is set for the argument voice_bit by the bitOR of SPU_0CH to SPU_23CH. The following value is set for "*status*" depending on the state of the streaming library.

**Table 13–44**

| State | Status |
|---|---|
| SPU_ST_PREPARE | SPU_ST_PREPARE |
| SPU_ST_PLAY | SPU_ST_PLAY |

### Return Value

The pointer towards the previously set callback function called at the completion of the data transfer in the stream preparation.

NULL is returned if no callback function has been previously set.

### Remarks

**See also:** SpuStTransfer (p.), SpuStSetTransferFinishedCallback (p.), SpuStSetStreamFinishedCallback (p.).

## SpuStSetStreamFinishedCallback

Sets the callback function called at the completion of each stream processing in the SPU streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**SpuStCallbackProc SpuStSetStreamFinishedCallback** (*SpuStCallbackProc callback_proc*)**;**
**SpuStCallbackProc *callback_proc** (*unsigned long voice_bit*, *long status*)

### Arguments

| | |
|---|---|
| *callback_proc* | The pointer towards the callback function called at the completion of each stream. |

### Explanation

Sets the callback function called at the completion of each stream in the SPU streaming.

When callback_proc is called, the value of the voices assigned for the stream of which processing is completed is set for the argument voice_bit by the bitOR of SPU_0CH to SPU_23CH. The following value is set for "*status*" depending on the state of the streaming library.

**Table 13–45**

| State | Status |
|---|---|
| SPU_ST_PLAY | SPU_ST_PLAY |
| SPU_ST_FINAL | SPU_ST_FINAL |

### Return Value

The pointer towards the previously set callback function called at the completion of each stream.

NULL is returned if no callback function has been previously set.

### Remarks

**See also:** SpuStTransfer (p.), SpuStSetPreparationFinishedCallback (p.), SpuStSetTransferFinishedCallback (p.).

## SpuStSetTransferFinishedCallback

Sets the callback function called at the completion of one transfer to the stream buffer for all the streams in the SPU streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**SpuStCallbackProc SpuStSetTransferFinishedCallback (***SpuStCallbackProc callback_proc***);**
**SpuStCallbackProc *callback_proc (***unsigned long voice_bit, long status***)**

### Arguments

| | |
|---|---|
| *callback_proc* | Pointer towards the callback function called at the completion of one transfer to the stream buffer for all the streams. |

### Explanation

Sets the callback function called at the completion of one transfer to the stream buffer for all the streams in the SPU streaming.

When callback_proc is called, the value of the voices assigned for the stream where one transfer to the stream buffer is completed is set for the argument voice_bit by the bitOR of SPU_0CH to SPU_23CH.

SPU_ST_PLAY is always set for "*status*".

### Return Value

The pointer towards the previously set callback function called at the completion of one transfer to the stream buffer for all the streams in the SPU streaming.

NULL is returned if no callback function has been previously set.

### Remarks

**See also:** SpuStTransfer (p.), SpuStSetPreparationFinishedCallback (p.), SpuStSetStreamFinishedCallback (p.).

## SpuStTransfer

Prepares for the stream, and provides the instruction for starting the stream in SPU streaming.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.2* | *7/31/96* |

### Syntax

**long SpuStTransfer** (**long** *flag*, **unsigned long** *voice_bit*);

### Arguments

| | |
|---|---|
| *flag* | Stream state flag |
| *voice_bit* | Streaming voices |

### Explanation

It prepares for the stream in the SPU streaming, and provides the instruction for starting it.

The voices where the stream is executed are set explicitly for *voice_bit* by the bitOR of SPU_0CH to SPU_23CH.

(1) STREAM STATE FLAG values: SPU_ST_PREPARE=Preparation

Carries out stream preparation according to the attributes of the SpuStEnv structure returned by SpuStInit().

The end of the preparation is determined by the callback function set by SpuStSetPreparationFinishedCallback.

(2) SPU_ST_PLAY=Start

The stream is started according to the attributes of the SpuStEnv structure returned by SpuStInit.

If the status of streaming is SPU_ST_PREPARE, the voice is keyed on promptly following analysis of the adequacy of voice_bit when this function is called.

If the status of SPU streaming is SPU_ST_TRANSFER, the transfer waits until processing is transferred to the latter part of the stream buffer of the currently processed streams. Consequently, the transfer is carried out with other stream processing when the latter part of the stream buffer is processed.

When one transfer to the stream buffer for all streams is completed, the callback function set by SpuStSetTransferFinishedCallback is called, and the attributes for the next transfer for each stream are set.

When a stream is completed, the callback function set by SpuStSetStreamFinishedCallback is called. (Precisely before the next transfer if other streams are processed.)

### Return value

| | |
|---|---|
| SPU_ST_NOT_AVAILABLE | SPU streaming is not available. SpuStInit() has not been called. |
| SPU_ST_INVALID_ARGUMENTS | The value of the arguments is not in the specification. |
| SPU_ST_WRONG_STATUS | SpuStTransfer is not accepted. |

The causes are:

- The current status is SPU_ST_FINAL without regard to flag.
- Flag is SPU_ST_PREPARE, and the current status is SPU_ST_PREPARE.
- Flag is SPU_ST_PLAY, and the current status is SPU_ST_IDLE.

| | |
|---|---|
| SPU_ST_ACCEPT | Processing is accepted. |

### Remarks

**See also:** SpuStInit (p.), SpuStSetPreparationFinishedCallback (p.), SpuStSetTransferFinishedCallback (p.), SpuStSetStreamFinishedCallback (p.).

## SpuStVoiceAttr*

SPU streaming voice attributes.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Structure

**typedef struct {**
  **char** *status*;
  **char** *pad1*;
  **char** *pad2*;
  **char** *pad3*;
  **long** *last_size*;
  **unsigned long** *buf_addr*;
  **unsigned long** *data_addr*;
**} SpuStVoiceAttr**

### Members

| | |
|---|---|
| *status* | Stream status |
| *pad1* | Padding |
| *pad2* | Padding |
| *pad3* | Padding |
| *last_size* | The size of final data transfer (last_size <= (size / 2)) |
| *buf_addr* | The start address of stream buffer |
| *data_addr* | The start address of stream in SPU RAM data in main RAM |

### Explanation

Holds each stream's attributes in the SPU streaming library.

### Remarks

**See also:** SpuStEnv (p. 13-7), SpuStInit (p. 13-113).

## SpuWrite

Transfers data from main memory to the sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuWrite**(**addr*, *size*)
**unsigned char** **addr*;
**unsigned long** *size*;

### Arguments

| | |
|---|---|
| *addr* | Pointer to transfer data start address in main memory |
| *size* | Transfer data size (in bytes) |

### Explanation

Transfers size bytes of data from main memory *addr* to the sound buffer

The main memory address addr storing the transfer data must fulfill the following conditions.

- It is an address of an allocated variable that is a global variable
- It is an address of an allocated variable that is in the heap and is allocated by a function such as malloc.

That is, it does not address a stack area (a variable = auto variable) declared in a function.

SpuWrite() does not perform sound buffer memory management, so real waveform data cannot be used if the user does not transfer to addresses which avoid the following areas.

- SPU decoded data transfer area: 0x0000-0xfff
- System reserved area: 0x1000-0x100f
- Addresses after the reverb work area offset (start) address

### Return value

Transferred data size.

If the specified data size is larger than 512 KB, the actual transferred size is returned.

### Remarks

**See also:** SpuRead (p.), SpuSetTransferStartAddr, SpuGetTransferStartAddr (p.), SpuWrite0 (p. 13-122), SpuWritePartly (p. 13-123).

## SpuWrite0

Clears sound buffer.

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuWrite0** (*size*)
**unsigned long** *size;*

### Arguments

| | |
|---|---|
| *size* | Clear area size (in bytes) |

### Explanation

Writes 0 in the sound buffer area.

This writing is done by DMA transfer, but is started synchronously.

The starting address of the area written is specified by SpuSetTransferStartAddr(), and its size is size.

### Return value

Returns the size of the area written with 0s.

If the data size set is larger than 512 KB, the actual written size is returned.

### Remarks

**See also:** SpuWrite (p. 13-121), SpuSetTransferStartAddr (p.).

## SpuWritePartly

Transfers data from main memory to the sound buffer (assuming the transfer is divided into sections).

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libspu.lib* | *Libspu.h* | *3.0* | *7/31/96* |

### Syntax

**unsigned long SpuWritePartly** (**addr*, *size*)
**unsigned char** **addr*;
**unsigned long** *size;*

### Arguments

| | |
|---|---|
| *addr* | Pointer to transfer data start address in main memory |
| *size* | Transfer data size (in bytes) |

### Explanation

Transfers data from main memory to the sound buffer.

The main memory address storing the transfer data must fulfill the following conditions.

- It is an address of an allocated variable that is a global variable
- It is an address of an allocated variable that is in the heap and is allocated by a function such as malloc.

That is, it does not address a stack area (a variable (= auto variable) declared in a function.

Data is transferred from the address specified in SpuSetTransferStartAddr(), and after completion of the transfer specified by size, the starting address is incremented by size, and stored internally.

Normally, in the case of continuous transfer, the size of each transfer must be a number divisible by 8. But when transferring the final block of a continuous transfer, the size need not be divisible by 8.

If SpuSetTransferStartAddr() is called during continuous transfer processing, correct continuous transfer is not guaranteed.

SpuWritePartly() does not perform sound buffer memory management, so real waveform data cannot be used if the user does not transfer to addresses which avoid the following areas.

- SPU decoded data transfer area: 0x0000-0xfff
- System reserved area: 0x1000-0x100f
- Addresses after the reverb work area offset (start) address

### Return value

Returns the size of the area written with 0s.

If the data size set is larger than 512 KB, the actual transferred size is returned.

### Remarks

**See also:** SpuWrite (p. 13-121), SpuRead (p. 13-60), SpuSetTransferStartAddr (p.), SpuGetTransferStartAddr (p. 13-28).

# Chapter 14: Serial Input/Output Library
# Table of Contents

## AddSIO

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsio.lib* | *Libsio.h* | *3.6* | *10/23/96* |

### Syntax

### Arguments

### Explanation

### Return values

### Remarks

See also:

## DelSIO

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsio.lib* | *Libsio.h* | *3.6* | *10/23/96* |

### Syntax

### Arguments

### Explanation

### Return values

### Remarks

**See also:**

## _sio_control

| Library | Header File | Introduced | Documentation Date |
|---|---|---|---|
| *Libsio.lib* | *Libsio.h* | *3.6* | *10/23/96* |

### Syntax

### Arguments

### Explanation

### Return values

### Remarks

See also: