# LibPSn00b Library Reference

*Runtime Library v0.15b*

PSn00bSDK
2018 - 2020 Meido-Tek Productions / PSn00bSDK Project

The PSn00bSDK tools and library programs fall under the terms of the Mozilla Public License. This copy of PSn00bSDK must include a copy of the Mozilla Public License, if not you must obtain a copy of this SDK with the license file intact.

As a quick summary of the Mozilla Public License; projects using MPL don't necessarily have to be under the MPL license and source disclosure is not required, only assets that are part of PSn00bSDK must be under MPL. If any derivative modifications have been made to PSn00bSDK that improves its functionality, such changes must be committed upstream to the main repository of PSn00bSDK under the terms of the MPL. It is also the only way to 'donate' to the project as the PSn00bSDK project does not accept any monetary donations.

For more details on the Mozilla Public License, you can read the full license at:
**https://www.mozilla.org/en-US/MPL/2.0/**

PSn00bSDK is 100% free to use for both hobbyist and commercial homebrew projects. Third party distribution of PSn00bSDK is probably not advisable as libraries and tools update regularly in its current state.

PSn00bSDK Github repository:
**https://github.com/Lameguy64/PSn00bSDK**

# Table of Contents

# About This Manual

The purpose of this manual is to describe all available LibPSn00b library functions, macros and structures that have been implement so far throughout the development of this project.

There are some plans to make a *LibPSn00b Library Overview* companion volume that further describes the structure, use and purpose of the libraries of LibPSn00b but is not yet being worked on due to limited available man power of the PSn00bSDK project as of the writing of this document.

## Changes Since Last Version

**CD-ROM Library:**

- Majority of library documented.

**Other:**

- Numerous typographical errors corrected which were mistakenly not logged.

## Related Documentation

Since an overview volume of the *LibPSn00b Runtime Library* is not yet made, the *Lameguy's PSX Programming Tutorial Series* is the best available substitute document for beginners alike for now. This can be found on the Tools & Resources page of the Lameguy64 website at **http://lameguy64.net/index.php?page=tools**.

The tutorial series covers both the Programmer's Tool/PsyQ SDK and PSn00bSDK and is also essential learning materials to those new to programming for the PSX.

**Note:** The Lameguy64 website additionally posts updates and current developments regarding PSn00bSDK and the LibPSn00b Runtime Libraries on occasion.

Nocash's PSX specs document may also be of great use, especially if you plan to go low level: **http://problemkaputt.de/psx-spx.htm**

## Documentation Credits

**Lead writer:** Lameguy64

# CD-ROM Library

## Table of Contents

## Structures

### CdlATV

CD-ROM attenuation parameters

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

**Structure**

```
typedef struct CdlATV
{
    u_char    val0;    CD to SPU L-to-L volume
    u_char    val1;    CD to SPU L-to-R volume
    u_char    val2;    CD to SPU R-to-R volume
    u_char    val3;    CD to SPU R-to-L volume
} CdlATV;
```

**Explanation**

This structure specifies parameters for the CD-ROM attenuation. Values must be of range 0 to 127.

The CD-ROM attenuation can be used to set the CD-ROM audio output to mono (0x40, 0x40, 0x40, 0x40) or reversed stereo (0x00, 0x80, 0x00, 0x80). It can also be used to play one of two stereo channels to both speakers.

The CD-ROM attenuation affects CD-DA and CD-XA audio.

**See also**

**CdMix**

## CdlDIR

CD-ROM directory query context handle

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *Yes* | *0.15b* | *02/28/2020* |

**Structure**

**typedef void* CdlDIR;**

**Explanation**

Used to store a directory context created by **CdOpenDir()**. An open context can then be used with **CdReadDir()** and closed with **CdCloseDir()**.

## CdlFILE

File entry structure

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Structure

```
typedef struct CdlFILE
{
    CdlLOC    loc;          CD-ROM position coordinates of file
    u_int     size;         Size of file in bytes
    char      name[16];     File name
} CdlFILE;
```

### Explanation

Used to store basic information of a file such as logical block location and size. Currently, **CdSearchFile()** is the only function that uses this struct but it will be used in directory listing functions that may be implemented in the future.

### See also

**CdSearchFile**

## CdlFILTER

Structure used to set CD-ROM XA filter

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/19/2019* |

### Structure

```
typedef struct CdlFILTER
{
    u_char     file;          File number to fetch (usually 1)
    u_char     chan;          Channel number (0 through 7)
    u_short    pad;           Padding
} CdlFILTER;
```

### Explanation

This structure is used to specify stream filter parameters for CD-ROM XA audio streaming using the **CdlSetfilter** command. This only affects CD-ROM XA audio streaming.

CD-ROM XA audio is normally comprised of up to 8 or more ADPCM compressed audio streams interleaved into one continuous stream of data. The data stream is normally read at 2x speed but only one of eight XA audio streams can be played at a time. The XA stream to play is specified by the **CdlSetfilter** command and this struct.

The CD-ROM XA filter can be changed during CD-ROM XA audio playback with zero audio interruption. This can be used to achieve dynamic music effects by switching to alternate versions of a theme to fit specific scenes seamlessly.

### See also

**CdControl**

## CdlLOC

CD-ROM positional coordinates

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Structure

```
typedef struct CdlLOC
{
    u_char    minute;    Minutes (BCD)
    u_char    second;    Seconds (BCD)
    u_char    sector;    Sector or frame (BCD)
    u_char    track;     Track number (not used)
} CdlLOC;
```

### Explanation

This structure is used to specify CD-ROM positional coordinates for **CdlSetloc**, **CdlReadN** and **CdlReadS** CD-ROM commands. Use **CdIntToPos()** to set parameters from a logical sector number.

### See also

**CdIntToPos CdControl**

# Functions

## CdAutoPauseCallback

Sets a callback function for auto pause

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *Yes* | *0.15b* | *12/18/2019* |

### Syntax

| | |
|---|---|
| **long *CdAutoPauseCallback(** | |
| **void(**func*)())** | Callback function |

### Explanation

Sets a callback function specified by **func*. Specifying 0 disables the callback.

The callback is executed when an auto pause interrupt occurs enabled by **CdlModeAP**. Auto pause occurs when CD Audio playback reaches the end of the audio track.

This can be used to easily loop CD audio automatically without any intervention in your software loop.

### Returns

Pointer to the last callback function set.

## CdCloseDir

Closes a directory context created by **CdOpenDir()**.

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *Yes* | *0.15b* | *02/28/2020* |

### Syntax

**void CdCloseDir(**
    **CdlDIR** **dir***)** Directory context

### Explanation

Closes an open directory context.

Behavior is undefined when closing an already closed directory context and may result in a crash.

### See also

**CdOpenDir**

## CdControl

Issues a control command to the CD-ROM controller

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *No* | *0.15b* | *12/12/2019* |

### Syntax

```
int CdControl(
    u_char    com,        Command value
    u_char    *param,     Command parameters
    u_char    *result)    Pointer of buffer to store result
```

### Description

Issues a CD-ROM command specified by *com* to the CD-ROM controller, waits for a command acknowledge interrupt (very fast) then returns. It will also issue parameters from *param* to the CD-ROM controller if the specified command takes parameters. Data returned from the CD-ROM controller is stored to *result* if the specified command returns data.

Because this function waits for an acknowledge interrupt from the CD-ROM controller, this function should not be used in a callback. Use **CdControlF()** instead.

Commands that are blocking require the use of **CdSync()** to wait for the command to fully complete.

**CD-ROM Control Commands:**

| Command | Value | Parameter | Blocking | Description |
|---|---|---|---|---|
| CdlNop | 0x01 | - | No | Also known as Getstat, usually used to acquire the current CD-ROM status. This is also used to signal the CD library if a disc change has occurred. |
| CdlSetloc | 0x02 | CdlLOC | No | Sets the seek target location. Actual seeking begins upon issuing **CdlSeekL**, **CdlSeekP**, **CdlPlay**, **CdlReadN** and **CdlReadS** commands. |
| CdlPlay | 0x03 | u_char | No | Begins CD Audio playback. CD mode must be set with **CdlModeDA** and **CdlSetMode** flags to work properly. **CdlModeAP** flag enables automatic pause at end of track. Parameter specifies the CD track to play and is optional (Note: some emulators do not support the optional track parameter). |
| CdlForward | 0x04 | - | No | Fast forward (only for CD Audio). |
| CdlBackward | 0x05 | - | No | Rewind (only for CD Audio). |
| CdlReadN | 0x06 | CdlLOC | No | Begin reading of data sectors. |
| CdlStandby | 0x07 | - | Yes | Also known as MotorOn, starts CD motor and stays idle. |
| CdlStop | 0x08 | - | Yes | Stops playback as well as the disc spinning. |
| CdlPause | 0x09 | - | Yes | Stops CD Audio/XA playback or data reading without stopping the disc. |

| | | | | |
|---|---|---|---|---|
| CdlInit | 0x0A | - | Yes | Initialize the CD-ROM controller. |
| CdlMute | 0x0B | - | No | Mutes CD audio (both DA and XA). |
| CdlDemute | 0x0C | - | No | Unmutes CD audio (both DA and XA). |
| CdlSetfilter | 0x0D | CdlFILTER | No | Sets XA audio filter. |
| CdlSetmode | 0x0E | u_char | No | Sets CD-ROM mode. |
| CdlGetparam | 0x0F | - | No | Returns current CD-ROM mode and file/channel filter settings. |
| CdlGetlocL | 0x10 | - | No | Returns current logical CD position as well as current mode and file/channel filter settings. |
| CdlGetlocP | 0x11 | - | No | Returns current physical CD position. |
| CdlSetsession (OC) | 0x12 | u_char | Yes | Seek to a specified session on a multi-session disc. |
| CdlGetTN | 0x13 | - | No | Get CD track count. |
| CdlGetTD | 0x14 | u_char | No | Get specified track position. |
| CdlSeekL | 0x15 | - | Yes | Logical seek to target position set by last CdlSetloc command. |
| CdlSeekP | 0x16 | - | Yes | Physical seek to target position set by last CdlSetloc command. |
| CdlTest (OC) | 0x19 | *varies* | Yes | Special test command not disclosed to official developers. |
| CdlReadS | 0x1B | CdlLOC | No | Begin reading sectors without error correction (for FMV streaming). |

**CD-ROM Return Values:**

| Command | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| CdlGetparam | stat | mode | 0 | file | channel | - | - | - |
| CdlGetlocL | amin | asec | aframe | mode | file | channel | sm | ci |
| CdlGetlocP | track | index | min | sec | frame | amin | asec | aframe |
| CdlGetTN | stat | first | last | - | - | - | - | - |
| CdlGetTD | stat | min | sec | - | - | - | - | - |

*All returned values are in BCD format.*

**Returns**

1 if the command was issued successfully. Otherwise 0 if a previously issued command has not yet finished processing.

**See also**

## CdControlB

Issues a CD-ROM command to the CD-ROM controller (non-blocking)

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | | |
|---|---|---|
| **int CdControlB(** | | |
| **u_char** | *com*, | Command value |
| **u_char** | **param*, | Command parameters |
| **u_char** | **result***)** | Pointer of buffer to store result |

### Explanation

This function works more or less the same as **CdControl()** but blocks for blocking commands until the command has completed.

Because this function waits for an acknowledge interrupt from the CD-ROM controller, this function should not be used in a callback. Use **CdControlF()** instead.

### See also

**CdControl**

## CdControlF

Issues a CD-ROM command to the CD-ROM controller (does not block)

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/19/2019* |

### Syntax

**int CdControlF(**

| | | |
|---|---|---|
| **u_char** | *com*, | Command value |
| **u_char** | **param***)** | Command parameters |

### Explanation

This function works more or less the same as **CdControl()** but it does not block even for the acknowledge interrupt from the CD-ROM controller. Since this function is non-blocking it can be used in a callback function.

When using this function in a callback, a maximum of two commands can be issued at once and only the first command can have parameters. This is because the CD-ROM controller can only queue up to two commands and the parameter FIFO is not cleared until the last command is acknowledged. But waiting for acknowledgment in a callback is not possible.

### See also

**CdControl**

## CdGetToc

Get CD-ROM TOC information

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

**int CdGetToc(**
**CdlLOC** **toc***)** Pointer to an array of **CdlLOC** entries

### Explanation

Retrieves all the track entries from the CD-ROM's table of contents (TOC). The function can return up to 99 track entries for it is the maximum number of tracks the CD-ROM standard supports.

The function can only retrieve the minutes and seconds of an audio track's position as the CD-ROM controller only returns the minutes and seconds of a track. This may result in the end of the previous track being played instead of the intended track to be played. This can be remedied by having a 2 second pregap on each CD Audio track.

### See also

**CdControl**

## CdGetSector

Get data from the CD-ROM sector buffered

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | | |
|---|---|---|
| **int CdGetSector(** | | |
| **void** | **madr,* | Pointer to store sector data to |
| **int** | *size***)** | Number of bytes to retrieve |

### Explanation

Retrieves sector data pending in the CD-ROM sector buffer and stores it to a buffer specified by **madr* when a data sector has been read. Uses DMA to get the sector data at high speed.

This function is a bit of a work in progress as *size* must be 2048 bytes or longer depending on the CD mode. If less than the desired sector size is retrieved a dummy read must be issued manually to clear the sector buffer.

### Returns

Always 1.

### See also

**CdReadyCallback**

## CdMode

Gets the last CD-ROM mode

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

**int CdMode(void)**

### Explanation

Returns the CD-ROM mode last set when issuing a **CdlSetMode** command. The function returns instantly as it returns a value from an internal variable.

Since the value is simply a copy of what was specified from the last **CdlSetMode** command, the mode value may become inaccurate if **CdlInit** or other commands that affect the CD-ROM mode were issued. This may be corrected in the future.

### Returns

CD-ROM mode last set.

## CdMix

Set CD-ROM mixer or attenuation

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

**int CdMix(**
**CdlATV** **vol***)** CD-ROM attenuation parameters.

### Explanation

Sets the CD-ROM attenuation parameters from a **CdlATV** struct specified by **vol*. The CD-ROM attenuation settings is different from the SPU CD-ROM volume.

### Returns

Always 1.

### See also

**CdlATV**

## CdStatus

Get the most recent CD-ROM status

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

**int CdStatus(void)**

### Explanation

Returns the CD-ROM status since the last command issued. The status value is updated by most CD-ROM commands.

To get the current CD-ROM status you can issue **CdlNop** commands at regular intervals to update the CD-ROM status this function returns.

### Returns

CD-ROM status from last comand issued.

### See also

**CdControl**

## CdPosToInt

Translates CD-ROM positional coordinates to a sector number

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | |
|---|---|
| **int CdPosToInt(** | |
| **CdlLOC** **p*) | Pointer to a **CdlLOC** struct. |

### Explanation

Translates the CD-ROM positional coordinates from a **CdlLOC** struct specified by *p* to a logical sector number. The translation takes the lead-in offset into account so the logical sector number begins at 0.

### Returns

Logical sector number.

## CdIntToPos

Translates a sector number to CD-ROM positional coordinates

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

**CdlLOC *CdIntToPos(**

| | | |
|---|---|---|
| **int** | *i*, | Logical sector number |
| **CdlLOC** | **p***)** | Pointer to a **CdlLOC** structure |

### Explanation

This function translates a logical sector number specified by *i* to CD-ROM positional coordinates stored to a **CdlLOC** struct specified by *p*. The translation takes the lead-in offset into account so the first logical sector begins at 0.

### Returns

Pointer to the specified **CdlLOC** struct.

## CdInit

Initializes the CD-ROM library

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *No* | *0.15b* | *12/12/2019* |

### Syntax

**int CdInit(**
    **int** *mode***)** Reserved (may be used in the future)

### Description

Initializes the CD-ROM subsystem which includes hooking the required IRQ handler, sets up internal variables of the CD-ROM library and attempts to initialize the CD-ROM controller. The *mode* parameter does nothing but may be used in future updates of this library.

This function must be called after **ResetGraph** and before any other CD-ROM library function that interfaces with the CD-ROM controller. This function may not be called twice as it may cause instability or would just crash.

### Returns

Always 1. May change in the future.

## CdIsoError

Retrieve CD-ROM ISO9660 parser status

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *Yes* | *0.15b* | *02/18/2020* |

### Syntax

**int CdIsoError()**

### Explanation

Returns the status of the file system parser from the last call of a file system related function, such as **CdSearchFile()**, **CdGetVolumeLabel()** and **CdOpenDir()**. Use this function to retrieve the exact error occurred when either of those functions fail.

### Returns

CD-ROM ISO9660 parser error code, as listed below.

| **Value** | **Description** |
|---|---|
| CdlIsoOkay | File system parser okay. |
| CdlIsoSeekError | Logical seek error occurred. May occur when attempting to query the file system while an Audio CD is inserted, which does not contain a file system. |
| CdlIsoReadError | Read error occurred while reading a file system descriptor. |
| CdlIsoInvalidFs | Disc does not contain a standard ISO9660 file system. |

## CdOpenDir

Open a directory on the CD-ROM file system

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *Yes* | *0.15b* | *02/28/2020* |

### Syntax

**CdlDIR* CdOpenDir(**
**const char*** *path***)** Directory path to open.

### Explanation

Opens a directory on the CD-ROM file system to read the contents of a directory.

A path name must use the backslash character (\) as the directory name separator (in C/C++, you must use double backslash as backslash is used to specify special characters in strings). The path must start with a backslash character and no device name (ie. \MYDIR1\MYDIR2 will work but not cdrom:\MYDIR1\MYDIR2).

The file system routines in libpsxcd can query directory paths of up to 128 characters.

The ISO9660 file system routines of libpsxcd does not support long file names as it only supports the original file descriptor format, which is limited to MS-DOS style 8.3 file names.

### Returns

Pointer of a **CdlDIR** context, NULL if an error occurred.

### See also

**CdReadDir CdCloseDir**

## CdRead

Read sectors from the CD-ROM

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | | |
|---|---|---|
| **int CdRead(** | | |
| **int** | *sectors,* | Number of sectors to read |
| **u_int** | **buf,* | Pointer to buffer to store sectors read |
| **int** | *mode***)** | CD-ROM mode for reading |

### Explanation

Reads a number sectors specified by sectors from the location set by the last **CdlSetloc** command and stores the data read to a buffer specified by *buf*. *mode* specifies the CD-ROM mode parameters to use for the read operation.

The size of the sector varies depending on the sector read mode specified by *mode*. For standard data sectors it is multiples of 2048 bytes. If **CdlModeSize0** is specified the sector size is 2328 bytes which includes the whole sector minus sync, adress, mode and sub header bytes. **CdlModeSize1** makes the sector size 2340 which is the entire sector minus sync bytes.

Ideally, **CdlModeSpeed** must be specified to read data sectors at double CD-ROM speed.

This function blocks very briefly to issue the necessary commands to start CD-ROM reading. To determine if reading has completed use **CdReadSync** or **CdReadCallback**.

### Returns

Always returns 0 even on errors. This may change in future versions.

## CdReadCallback

Sets a callback function for read completion

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/19/2019* |

### Syntax

| | |
|---|---|
| **u_int CdReadCallback(** | |
| **CdlCB** *func***)** | Callback function |
| | |
| **void (****func***)(int** *status*, | CD-ROM status |
| **u_char** **result***)** | Pointer to a result buffer |

### Explanation

Works much the same as **CdSyncCallback()** but for **CdRead()**. Sets a callback with the specified function *func*. The callback is executed whenever a read operation initiated by **CdRead()** has completed.

*status* is the CD-ROM status from the command that has completed processing. **result* points to a read result buffer.

### See also

**CdRead**

## CdReadDir

Read a directory entry from an open directory context

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *02/28/2020* |

### Syntax

**int CdReadDir(**

| | | |
|---|---|---|
| **CdlDIR** | **dir,* | Open directory context (from **CdOpenDir()**) |
| **CdlFILE** | **file***)** | Pointer to a **CdlFILE** struct |

### Explanation

Retrieves a file entry from an open directory context and stores it to a **CdlFILE** struct specified by *file*. Repeated calls of this function retrieves the next directory entry available until there are no more directory entries that follow.

### Returns

1 if there are proceeding directory entries that follow, otherwise 0.

### See also

**CdOpenDir**

## CdReadSync

Waits for CD-ROM read completion or returns read status

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *No* | *0.15b* | *12/19/2019* |

### Syntax

| | | |
|---|---|---|
| **int CdReadSync(** | | |
| **int** | *mode,* | Mode |
| **u_char** | **result***)** | Pointer to store most recent CD-ROM status |

### Explanation

This function works more or less like **CdSync()** but for **CdRead()**. If *mode* is zero the function blocks if **CdRead()** was issued earlier until reading has completed. If mode is non-zero the function returns a read status value.

A buffer specified by *result* will be set with the most recent CD-ROM status value from the last read issued.

### Returns

Command status is returned as one of the following definitions:

| | |
|---|---|
| CdlComplete | Read completed. |
| CdlNoIntr | No interrupt, command busy. |
| CdlDiskError | CD-ROM error occurred. |

### See also

**CdRead**

## CdReadyCallback

Sets a callback function

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | |
|---|---|
| **long CdReadyCallback(** | |
| **CdlCB** *func***)** | Callback function |
| **void (****func***)(int** *status*, | CD-ROM status |
| **u_char** **result***)** | Pointer to a result buffer |

### Explanation

Sets a callback with the specified function *func*. The callback is executed whenever there's an incoming data sector from the CD-ROM controller during **CdlReadN** or **CdlReadS**. The pending sector data can be retrieved using **CdGetSector()**.

*status* is the CD-ROM status from the command that has completed processing. **result* corresponds to the **result* parameter on **CdControl()/CdControlB()** and returns the pointer to the buffer last set with that function.

This callback should be used in conjunction with **CdRead()** because that function uses this callback hook internally. The previously set callback is retained upon read completion however.

### Returns

Pointer to last callback function set.

### See also

**CdControl CdControlB**

## CdSearchFile

Locates a file in the CD-ROM file system

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *No* | *0.15b* | *12/19/2019* |

### Syntax

| | |
|---|---|
| **CdlFILE *CdSearchFile(** | |
| **CdlFILE** **loc,* | Pointer to a **CdlLOC** struct to store file information |
| **const char** **filename***)** | Path and name of file to locate |

### Explanation

Searches a file specified by *filename* by path and name in the CD-ROM file system and returns information of the file if found. The file information acquired will be stored to *loc*.

Directories must be separated with backslashes (\) and a leading backslash is optional and paths must reference from the root directory. File version identifier (;1) at the end of the file name is also optional. File and directory names are case insensitive.

The ISO9660 file system routines of libpsxcd does not support long file names as it only supports the original file descriptor format, which is limited to MS-DOS style 8.3 file names.

Upon calling this function for the first time, the ISO descriptor of the disc is read and the whole path table is cached into memory. Next the directory descriptor of the particular directory specified is loaded and cached to locate the file specified. The directory descriptor is kept in memory as long as the consecutive files to be searched are stored in the same directory until a file in another directory is to be searched. On which the directory descriptor is unloaded and a new directory descriptor is read from the disc and cached. Therefore, locating files in the same directory is faster as the relevant directory descriptor is already in memory and no disc reads are issued.

As of version 1.15b of the PSn00bSDK run-time library, there is currently no official method to tell CdSearchFile to re-read the CD-ROM ISO descriptor and path table for disc changes. This will be resolved in a future release.

### Returns

Pointer to the specified **CdlFILE** struct. Otherwise NULL is returned when the file is not found.

## CdSync

Wait for blocking command or blocking status

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | | |
|---|---|---|
| **int CdSync(** | | |
| **int** | *mode*, | Mode |
| **u_char** | **result***)** | Pointer to store most recent CD-ROM status |

### Explanation

If *mode* is zero the function blocks if a blocking command was issued earlier until the command has finished. If mode is non-zero the function returns a command status value.

A buffer specified by *result* will be set with the most recent CD-ROM status value from the last command issued.

### Returns

Command status is returned as one of the following definitions:

| | |
|---|---|
| CdlComplete | Command completed. |
| CdlNoIntr | No interrupt, command busy. |
| CdlDiskError | CD-ROM error occurred. |

### See also

**CdControl**

## CdSyncCallback

Sets a callback function

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | |
|---|---|
| **u_int CdSyncCallback(** | |
| **CdlCB** *func***)** | Callback function |
| | |
| **void (****func***)(int** *status*, | CD-ROM status |
| **u_char** **result***)** | Pointer to a result buffer |

### Explanation

Sets a callback with the specified function *func*. The callback is executed whenever a blocking command has completed.

*status* is the CD-ROM status from the command that has completed processing. **result* corresponds to the **result* parameter on **CdControl()/CdControlB()** and returns the pointer to the buffer last set with that function.

### Returns

Pointer to last callback function set.

### See also

**CdControl CdControlB CdSync**

## Macros

### btoi

Translates a BCD format value to decimal

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

**Syntax**

**btoi(**
*b***)** BCD format value

**Explanation**

Translates a specified value in BCD format (ie. 32/0x20 = 20) into a decimal integer, as the CD-ROM controller returns integer values only in BCD format.

## itob

Translates a decimal value to BCD

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxcd.a* | *psxcd.h* | *No* | *0.15b* | *12/18/2019* |

### Syntax

| | |
|---|---|
| **itob(** | |
| ***i*)** | Decimal value |

### Explanation

Translates a decimal integer into a BCD format value (ie. 20 = 32/0x20), as the CD-ROM controller only accepts values in BCD format.

# Graphics Library

## Chapter Contents

## Overview

The graphics library provides functions for initializing and controlling the GPU hardware as well as various structures and macros for preparing graphics primitives to be drawn by the GPU. This library does not provide functions for 3D graphics processing, the Geometry Library (psxgte) provides such functions instead.

This library also provides a global ISR handler which other libraries depend on for handling interrupts and is installed to the kernel by ResetGraph(). Even if you don’t plan to do any graphics, it is highly recommended to call ResetGraph() at the beginning of your program.

## Structures

### DISPENV

Display environment structure

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

**Structure**

```
typedef struct {
    RECT    disp;       Display coordinates (framebuffer position and resolution)
    RECT    screen;     Screen coordinates (picture position and size)
    char    isinter;    Interlace flag (0: non-interlace, 1: interlace)
    char    isrgb24;    RGB24 color mode (0: 16-bit color mode, 1: 24-bit color mode)
    short   pad;        Padding
} DISPENV;
```

**Explanation**

This structure specifies the display attributes to apply to the GPU using PutDispEnv().

*The disp* element specifies both the offset of the framebuffer area to be displayed (*disp.x, disp.y*) and display resolution. Valid horizontal resolutions (for *disp.w*) are 256, 320, 384, 512 and 640 and vertical resolutions (for *disp.h*) are 240 and 480 for NTSC standard and 256 and 512 for PAL standard. The display resolution also determines the size of the rectangular area on the framebuffer to be displayed. If the display area exceeds the framebuffer area the picture would simply wrap around to the other side of the framebuffer.

Apparently the GPU is capable of outputting 272 vertical lines in PAL standard even if you have the vertical resolution set to 256. This is yet to be investigated further.

The *screen* element specifies the position (*screen.x*, *screen.y*) and size (*screen.w*, *screen.h*) of the picture displayed on the TV screen. A position of (0, 0) is the base position of the picture and if the picture size is set to (0, 0), default size values are used based on the resolution specified by the *disp* element. Specifying values that are lower or greater than the resolution specified by *disp* can be used to achieve custom resolutions but the hardware will not scale the pixels, it merely just crops or extends what is being shown.

The *isinter* flag specifies if the video signal should be interlaced. This flag must be set when using a vertical resolution of 480 or 512 pixel lines, otherwise, only the even lines would be displayed or a strange video collapse effect will occur (and no, the GPU hardware cannot output 480p at all). Interlace can be set for 240 and 256 line modes but it would introduce unnecessary jitter though it may help improve compatibility with HDTVs and video capture devices that expect an interlace jitter signal.

The *isrgb24* flag specifies 24-bit true-color mode and expands the display area by 1.5x horizontally to accommodate the additional bytes needed for RGB24 pixels. This mode can’t be used for real-time graphics as the GPU only renders at 16-bit color so 24-bit mode is most useful for FMV sequences or displaying graphic illustrations from MDEC compressed image data (after decompression).

**See also**

**PutDispEnv**

## DRAWENV

Drawing environment structures

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Structure

**typedef struct {**

| | | |
|---|---|---|
| **RECT** | *clip;* | Drawing area in framebuffer within (0, 0) – (1023, 511) |
| **short** | *ofs[2];* | GPU drawing offset (x, y) |
| **RECT** | *tw;* | Initial texture page window coordinates |
| **u_short** | *tpage;* | Initial texture page (see getTPage()) |
| **u_char** | *dtd;* | Dither processing (0: no dithering, 1: dithered) |
| **u_char** | *dfe;* | Allow drawing to displayed area (0: don’t draw to display area, 1: draw) |
| **u_char** | *isbg;* | Draw area clear on environment set (0: no clear, 1: clear) |
| **u_char** | *r0,g0,b0;* | Draw area clear color |
| **DR_ENV** | *dr_env;* | Drawing environment buffer (reserved) |

**} DRAWENV;**

### Explanation

This structure specifies the drawing attributes to apply to the GPU using PutDrawEnv().

The *clip* element specifies the rectangular area of the framebuffer that graphics primitives will be drawn to. The drawing area can be of any arbitrary size as long as it is within the framebuffer area.

The *ofs[]* element specifies the X,Y coordinates of the GPU offset which is the position where a coordinate of (0,0) will originate from. The coordinates specified are relative to the *clip* area coordinates.

The *tw* element specifies the texture window size and offset of the texture page. Currently that functionality is not yet implemented in PSn00bSDK so this element does nothing.

The *tpage* element specifies the initial texture page value to set to the GPU. A texture page can be easily calculated using getTPage() and the texture page can be changed mid-drawing using the DR_TPAGE packet.

The *dtd* element specifies if dither processing is enabled or not. The dither processing bit is merged with the specified texture page value and could be disabled if a DR_TPAGE primitive was processed without the dither processing bit set.

The *dfe* element specifies if drawing should be blocked if the area is occupied by a display area. This is normally set to zero since most page flipping setups usually draw to an area not visible to the display and is mandatory for hi-res modes as it would allow the GPU to only draw on rows that are not being displayed, allowing for a pseudo double buffered setup. Setting this to non-zero would allow drawing in a display area as well as draw on both fields in hi-res modes which might be useful for static menu screens in hi-res.

The *isbg* element specifies if the drawing area should be cleared when this structure is applied using PutDrawEnv(), recommended for instances where the screen is constantly being updated. The clear color is specified using the *r0,g0,b0* elements.

The *dr_env* element is a reserved element used as a buffer by PutDrawEnv(). The DR_ENV structure can be used as a primitive packet to change the drawing environment mid-drawing for split-screen setups or off-screen render-to-texture tricks for example.

### Work in progress

The *tw* element has no effect to the drawing environment as of version 0.09b.

### See also

### PutDrawEnv

## RECT

Defines a rectangular area

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *12/21/2018* |

**Structure**

```
typedef struct {
    short       x,y;          Top left coordinates of the rectangular area
    short       w,h;          Width and height of the rectangular area
} RECT;
```

**Explanation**

Used to define a rectangular area in various structures and functions.

## TIM_IMAGE

Texture Image parameters

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *02/02/2019* |

### Structure

```
typedef struct {
    u_int    mode;      Image mode (bit 0-3: color depth, bit 4: CLUT flag)
    RECT     *crect;    Pointer to CLUT rectangle coordinates
    u_int    *caddr;    Pointer to CLUT data (or NULL if no CLUT)
    RECT     *prect;    Pointer to pixel data rectangle coordinates
    u_int    *paddr;    Pointer to pixel data
} TIM_IMAGE;
```

### Explanation

Used to store texture image parameters from a TIM file with **GetTimInfo**. The *crect, caddr, prect* and *paddr* elements can be referenced directly to access TIM coordinates and data easily.

### See also

**GetTimInfo**

## Structures (Primitives)

### DR_MASK

Mask mode primitive

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *Yes* | *0.01b* | *07/17/2019* |

**Structure**

```
typedef struct {
    u_int       tag;            Pointer to next primitive + length of this packet
    u_int       code[1];        Drawing mask primitive code
} DR_MASK;
```

**Explanation**

Sets the drawing mask setting of the GPU, a limited implementation of stencil masks.

**See also**

**setDrawMask**

## DR_TPAGE

Texture page primitive

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/16/2019* |

### Structure

```
typedef struct {
    u_int    tag;        Pointer to next primitive + length of this packet
    u_int    code[1];    Texture page primitive code
} DR_TPAGE;
```

### Explanation

A texture page primitive, used to change the current Tpage of the GPU mid-drawing.

Used alongside primitives that lack a Tpage field, such as **SPRT**, **SPRT_8 and SPRT_16** primitives, and for setting the blend operator of untextured primitives, such as **TILE**, **TILE_1, TILE_8, TILE_16**, **POLY_F3, POLY_F4**, **POLY_G3, and POLY_G4** primitives, that have been set for semi-transparency.

### See also

**setDrawTPage setDrawTPageVal**

## LINE_F2, LINE_F3, LINE_F4

2-point, 3-point and 4-point solid colored line primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *07/16/2019* |

### Structure

```
typedef struct {
    u_int     tag;              Pointer to next primitive + length of this packet
    u_char    r0,g0,b0,code;    RGB color + primitive code
    short     x0,y0;            Screen coordinates 0
    short     x1,y1;            Screen coordinates 1
} LINE_F2;

typedef struct {
    u_int     tag;              Pointer to next primitive + length of this packet
    u_char    r0,g0,b0,code;    RGB color + primitive code
    short     x0,y0;            Screen coordinates 0
    short     x1,y1;            Screen coordinates 1
    short     x2,y2;            Screen coordinates 2
    u_int     pad;              Terminator value (usually 0x55555555)
} LINE_F3;

typedef struct {
    u_int     tag;              Pointer to next primitive + length of this packet
    u_char    r0,g0,b0,code;    RGB color + primitive code
    short     x0,y0;            Screen coordinates 0
    short     x1,y1;            Screen coordinates 1
    short     x2,y2;            Screen coordinates 2
    short     x3,y3;            Screen coordinates 3
    u_int     pad;              Terminator value (usually 0x55555555)
} LINE_F4;
```

### Explanation

LINE_F2 draws a solid colored 2-point line between (*x0*, *y0*) – (*x1*, *y1*) with color specified by (*r0, g0, b0*).

LINE_F3 draws a solid colored 3-point line around (*x0*, *y0*) – (*x1*, *y1*) – (*x2, y2*) with color specified by (*r0, g0, b0*).

LINE_F4 draws a solid colored 4-point line around (*x0*, *y0*) – (*x1, y1*) – (*x2, y2*) – (*x3, y3*) with color specified by (*r0, g0, b0*).

### See also

**setLineF2 setLineF3 setLineF4**

## LINE_G2, LINE_G3, LINE_G4

2-point, 3-point and 4-point shaded line primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *07/16/2019* |

### Structures

```
typedef struct {
    u_int    tag;            Pointer to next primitive + length of packet
    u_char   r0,g0,b0,code;  RGB color 0 + primitive code
    short    x0,y0;          Screen coordinates 0
    u_char   r1,g1,b1,p1;    RGB color 1 + padding
    short    x1,y1;          Screen coordinates 0
} LINE_G2;

typedef struct {
    u_int    tag;            Pointer to next primitive + length of packet
    u_char   r0,g0,b0,code;  RGB color 0 + primitive code
    short    x0,y0;          Screen coordinates 0
    u_char   r1,g1,b1,p1;    RGB color 1 + padding
    short    x1,y1;          Screen coordinates 1
    u_char   r2,g2,b2,p2;    RGB color 2 + padding
    short    x2,y2;          Screen coordinates 2
    u_int    pad;            Terminator value (usually 0x55555555)
} LINE_G3;

typedef struct {
    u_int    tag;            Pointer to next primitive + length of packet
    u_char   r0,g0,b0,code;  RGB color 0 + primitive code
    short    x0,y0;          Screen coordinates 0
    u_char   r1,g1,b1,p1;    RGB color 1 + padding
    short    x1,y1;          Screen coordinates 1
    u_char   r2,g2,b2,p2;    RGB color 2 + padding
    short    x2,y2;          Screen coordinates 2
    u_char   r3,g3,b3,p3;    RGB color 3 + padding
    short    x3,y3;          Screen coordinates 3
    u_int    pad;            Terminator value (usually 0x55555555)
} LINE_G4;
```

### Explanation

LINE_F2 draws a solid colored 2-point line between (*x0*, *y0*) – (*x1*, *y1*) with color specified by (*r0, g0, b0*) – (*r1, g1, b1*).

LINE_F3 draws a solid colored 3-point line around (*x0*, *y0*) – (*x1, y1*) – (*x2, y2*) with color specified by (*r0, g0, b0*) – (*r1, g1, b1*) – (*r2, g2, b2*).

LINE_F4 draws a solid colored 4-point line around (*x0*, *y0*) – (*x1, y1*) – (*x2, y2*) – (*x3, y3*) with color specified by (*r0, g0, b0*) – (*r1, g1, b1*) – (*r2, g2, b2*) – (*r3, g3, b3*).

## P_TAG

Generic primitive header

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

**Structure**

| | | |
|---|---|---|
| **typedef struct {** | | |
| **u_int** | *addr:24;* | Next primitive address |
| **u_int** | *len:8;* | Primitive length (in words) |
| **u_char** | *r,g,b;* | Primitive color |
| **u_char** | *code;* | Primitive code |
| **} P_TAG;** | | |

**Explanation**

Normally used in various primitive preparation macros and the **addPrim** macro.

## POLY_F3, POLY_F4

3-point and 4-point, untextured, flat shaded polygon primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Structure

```
typedef struct {
    u_int     tag;              Pointer tag to primitive + packet length
    u_char    r0,g0,b0,code;    RGB color + primitive code
    short     x0,y0;            Screen coordinates 0
    short     x1,y1;            Screen coordinates 1
    short     x2,y2;            Screen coordinates 2
} POLY_F3;

typedef struct {
    u_int     tag;              Pointer tag to primitive + packet length
    u_char    r0,g0,b0,code;    RGB color + primitive code
    short     x0,y0;            Screen coordinates 0
    short     x1,y1;            Screen coordinates 1
    short     x2,y2;            Screen coordinates 2
    short     x3,y3;            Screen coordinates 3
} POLY_F4;
```

### Explanation

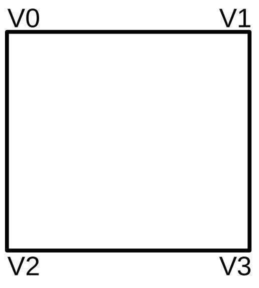
POLY_F3 draws a 3-point flat shaded, untextured polygon to screen coordinates (*x0,y0*) – (*x1,y1*) – (*x2,y2*).

POLY_F4 draws a 4-point flat shaded, untextured polygon to screen coordinates (*x0,y0*) – (*x1,y1) –* (*x2,y2*) – (*x3,y3*).

Elements *r0, g0, b0* specifies the color of the primitive.

Use **setPolyF3** and **setPolyF4** macros respectively to initialize the primitive before adding it to an ordering table.

The following figure describes the vertex order for 4-point polygons:



## POLY_FT3, POLY_FT4

3-point and 4-point, textured, flat shaded polygon primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Structure

```
typedef struct {
    u_int    tag;              Pointer tag to primitive + packet length
    u_char   r0,g0,b0,code;    RGB color + primitive code
    short    x0,y0;            Screen coordinates 0
    u_char   u0,v0;            Texture coordinates 0
    u_short  clut;             Texture CLUT ID
    short    x1,y1;            Screen coordinates 1
    u_char   u1,v1;            Texture coordinates 1
    u_short  tpage;            Texture page
    short    x2,y2;            Screen coordinates 2
    u_char   u2,v2;            Texture coordinates 2
    u_short  pad;              Padding
} POLY_FT3;

typedef struct {
    u_int    tag;              Pointer tag to primitive + packet length
    u_char   r0,g0,b0,code;    RGB color + primitive code
    short    x0,y0;            Screen coordinates 0
    u_char   u0,v0;            Texture coordinates 0
    u_short  clut;             Texture CLUT ID
    short    x1,y1;            Screen coordinates 1
    u_char   u1,v1;            Texture coordinates 1
    u_short  tpage;            Texture page
    short    x2,y2;            Screen coordinates 2
    u_char   u2,v2;            Texture coordinates 2
    u_short  pad0;             Padding
    short    x3,y3;            Screen coordinates 3
    u_char   u3,v3;            Texture coordinates 3
    u_short  pad1;             Padding
} POLY_FT4;
```

### Explanation

POLY_FT3 draws a 3-point flat shaded, textured polygon to screen coordinates (*x0,y0*) – (*x1,y1*) – (*x2,y2*).

POLY_FT4 draws a 4-point flat shaded, textured polygon to screen coordinates (*x0,y0*) – (*x1,y1)* – (*x2,y2*) – (*x3,y3*).

Elements (*u0,v0*), (*u1,v1),* (*u2,v2*) and (*u3,v3*) specify the texture coordinates within the texture page specified by *tpage*. Texture CLUT ID is specified by the *clut* element.

Elements *r0, g0, b0* specifies the color of the primitive.

Use **setPolyFT3** and **setPolyFT4** macros respectively to initialize the primitive before adding it to an ordering table.

See **POLY_F3, POLY_F4** for a visual figure of the vertex order for 4-point polygons.

## POLY_G3, POLY_G4

3-point and 4-point, untextured, gouraud shaded polygon primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Structure

```
typedef struct {
    u_int     tag;                Pointer tag to primitive + packet length
    u_char    r0,g0,b0,code;      RGB color 0 + code
    short     x0,y0;              Screen coordinates 0
    u_char    r1,g1,b1,pad0;      RGB color 1
    short     x1,y1;              Screen coordinates 1
    u_char    r2,g2,b2,pad1;      RGB color 2
    short     x2,y2;              Screen coordinates 2
} POLY_G3;

typedef struct {
    u_int     tag;                Pointer tag to primitive + packet length
    u_char    r0,g0,b0,code;      RGB color 0 + code
    short     x0,y0;              Screen coordinates 0
    u_char    r1,g1,b1,pad0;      RGB color 1 + padding
    short     x1,y1;              Screen coordinates 1
    u_char    r2,g2,b2,pad1;      RGB color 2 + padding
    short     x2,y2;              Screen coordinates 2
    u_char    r3,g3,b3,pad2;      RGB color 3 + padding
    short     x3,y3;              Screen coordinates 3
} POLY_G4;
```

### Explanation

POLY_G3 draws a 3-point flat shaded, textured polygon to screen coordinates (*x0,y0*) – (*x1,y1*) – (*x2,y2*).

POLY_G4 draws a 4-point flat shaded, textured polygon to screen coordinates (*x0,y0*) – (*x1,y1) –* (*x2,y2*) – (*x3,y3*).

Elements (*r0,g0,b0*), (*r1,g1,b1*), (*r2,g2,b2*) and (*r3,g3,b3*) specifies the color of the primitive for each point.

Use **setPolyG3** and **setPolyG4** macros respectively to initialize the primitive before adding it to an ordering table.

See **POLY_F3, POLY_F4** for a visual figure of the vertex order for 4-point polygons.

## POLY_GT3, POLY_GT4

3-point and 4-point, textured, gouraud shaded polygon primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Structure

```
typedef struct {
    u_int     tag;              Pointer tag to primitive + packet length
    u_char    r0,g0,b0,code;    RGB color 0 + code
    short     x0,y0;            Screen coordinates 0
    u_char    u0,v0;            Texture coordinates 0
    u_short   clut;             Texture CLUT ID
    u_char    r1,g1,b1,pad0;    RGB color 1
    short     x1,y1;            Screen coordinates 1
    u_char    u1,v1;            Texture coordinates 1
    u_short   tpage;            Texture page ID
    u_char    r2,g2,b2,pad1;    RGB color 2
    short     x2,y2;            Screen coordinates 2
    u_char    u2,v2;            Texture coordinates 2
    u_short   pad2;             Padding
} POLY_GT3;

typedef struct {
    u_int     tag;              Pointer tag to primitive + packet length
    u_char    r0,g0,b0,code;    RGB color 0 + code
    short     x0,y0;            Screen coordinates 0
    u_char    u0,v0;            Texture coordinates 0
    u_short   clut;             Texture CLUT ID
    u_char    r1,g1,b1,pad0;    RGB color 1
    short     x1,y1;            Screen coordinates 1
    u_char    u1,v1;            Texture coordinates 1
    u_short   tpage;            Texture page ID
    u_char    r2,g2,b2,pad1;    RGB color 2
    short     x2,y2;            Screen coordinates 2
    u_char    u2,v2;            Texture coordinates 2
    u_short   pad2;             Padding
    u_char    r3,g3,b3,pad3;    RGB color 3
    short     x3,y3;            Screen coordinates 3
    u_char    u3,v3;            Texture coordinates 3
    u_short   pad4;             Padding
} POLY_GT4;
```

### Explanation

POLY_GT3 draws a 3-point gouraud shaded, textured polygon to screen coordinates (*x0,y0*) – (*x1,y1*) – (*x2,y2*).

POLY_GT4 draws a 4-point gouraud shaded, textured polygon to screen coordinates (*x0,y0*) – (*x1,y1) –* (*x2,y2*) – (*x3,y3*).

Elements (*u0,v0*), (*u1,v1),* (*u2,v2*) and (*u3,v3*) specify the texture coordinates within the texture page specified by *tpage*. Texture CLUT ID for color-index textures is specified by the *clut* element.

Elements (*r0,g0,b0*)*,* (*r1,g1,b1*), (*r2,g2,b2*) and (*r3,g3,b3*) specifies the color of the primitive for each point.

Use **setPolyGT3** and **setPolyGT4** macros respectively to initialize the primitive before adding it to an ordering table.

See **POLY_F3, POLY_F4** for a visual figure of the vertex order for 4-point polygons.

## SPRT

Any-size textured sprite

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *05/23/2019* |

**Structure**

```
typedef struct {
    u_int     tag;              Pointer tag to next primitive packet
    u_char    r0,g0,b0,code;    RGB color of sprite + packet code
    short     x0,y0;            Position of sprite
    u_char    u0,v0;            Sprite texture coordinates within texture page. u0 must be a multiple of 2
    u_short   clut;             Sprite texture CLUT ID (see getClut)
    u_short   w,h;              Sprite size (w must be a multiple of 2)
} SPRT;
```

**Explanation**

Draws a textured sprite primitive of any defined size, draws faster than POLY_FT4 but lacks the authority for scaling and rotation.

If you use a sprite size greater than 256x256 (or the size of the texture window), the texture will simply repeat.

Because the SPRT primitive has no element to specify a texture page, a DR_TPAGE primitive can be used to work around that limitation. In order for the primitive to be effective, it must be added to the ordering table after the SPRT primitive has been sorted and both primitives must be added to the same element of the ordering table.

Use **setSprt** to initialize the primitive before adding it to the ordering table.

## SPRT_8, SPRT_16

Fixed size 8 x 8 or 16 x 16 textured sprite

| Library | Header | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *05/23/2019* |

### Structure

```
typedef struct {
    u_int       tag;                Pointer tag to next primitive packet
    u_char      r0,g0,b0,code;      RGB color of sprite + primitive code
    short       x0,y0;              Position of sprite (top-left coordinates)
    u_char      u0,v0;              Sprite texture coordinates within texture page, u0 must be a multiple of 2
    u_short     clut;               Sprite texture CLUT ID (see getClut)
} SPRT_8;

typedef struct {
    u_int       tag;                Pointer tag to next primitive packet
    u_char      r0,g0,b0,code;      RGB color of sprite + primitive code
    short       x0,y0;              Position of sprite (top-left coordinates)
    u_char      u0,v0;              Sprite texture coordinates within texture page, u0 must be a multiple of 2
    u_short     clut;               Sprite texture CLUT ID (see getClut)
} SPRT_16;
```

### Explanation

Draws a fixed size 8 x 8 or 16 x 16 pixel textured sprite, supposedly faster than **SPRT**.

Much like **SPRT** it has no texture page element so a DR_TPAGE primitive must be added to the ordering table after the **SPRT** primitive to specify the desired texture page value.

Use **setSprt8** and **setSprt16** respectively to initialize the packet before adding it to an ordering table.

## TILE

Any size flat colored sprite

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *05/23/2019* |

**Structure**

```
typedef struct {
    u_int     tag;              Pointer tag to next primitive packet
    u_char    r0,g0,b0,code;    RGB color of tile + packet code
    short     x0,y0;            Position of tile (top-left coordinate)
    short     w,h;              Size of tile in pixels
} TILE;
```

**Explanation**

Draws a flat colored sprite of specified size.

Use **setTile** to initialize the packet before adding it to an ordering table.

## TILE_1, TILE_8, TILE_16

Fixed size 1 x 1, 8 x 8 and 16 x 16 colored sprites.

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.1b* | *05/23/2019* |

**Structure**

```
typedef struct {
    u_int     tag;              Pointer tag to next primitive packet
    u_char    r0,g0,b0,code;    RGB color of tile + packet code
    short     x0,y0;            Position of tile (top-left coordinates)
} TILE_1;

typedef struct {
    u_int     tag;              Pointer tag to next primitive packet
    u_char    r0,g0,b0,code;    RGB color of tile + packet code
    short     x0,y0;            Position of tile (top-left coordinates)
} TILE_8;

typedef struct {
    u_int     tag;              Pointer tag to next primitive packet
    u_char    r0,g0,b0,code;    RGB color of tile + packet code
    short     x0,y0;            Position of tile (top-left coordinates)
} TILE_16;
```

**Explanation**

Draws a fixed size 1 x 1, 8 x 8 or 16 x 16 flat colored sprite.

Use **setTile1**, **setTile8**, **setTile16** to initialize the packet before adding it to an ordering table.

# Functions

## AddPrim

Non macro version of **addPrim**

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Syntax

| | |
|---|---|
| **void AddPrim(** | |
| **u_int** **ot,* | Pointer to an ordering table element |
| **void** **p*) | Pointer to a primitive packet |

### Explanation

Links a primitive packet to an ordering table element by setting the value from the specified table element to the primitive packet's tag element (with the size byte retained) and the pointer to the packet is set to the specified table element.

It is recommended to generate primitive packets in a global buffer to ensure that they do not get overwritten when the GPU gets around to processing the primitive (ie. If you allocate the primitive as a local variable in a function, it may have been overwritten when the GPU gets to draw it).

A common misconception among many PS1 homebrew programmers is they sometimes believe only a single primitive packet can be added to each ordering table element. This is false because adding another primitive to an ordering table element that already has a primitive concatenates to the chain, not replace the element.

Therefore, an ordering table length of 4 to 8 elements is usually enough for purely 2D projects. Higher ordering table sizes are recommended for projects featuring 3D visuals.

### See also

**ClearOTagR DrawOTag**

## ClearOTagR

Initializes an array to an empty ordering table (reverse order)

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *12/21/2018* |

### Syntax

| | |
|---|---|
| **void ClearOTagR(** | |
| **unsigned int** **ot,* | Pointer to an array to initialize into a linked list |
| **int** *n***)** | Number of array elements |

### Explanation

Initializes an array of *n* elements specified by **ot* into a linked list to use as an ordering table. An ordering table consists of an array of pointers that point from one entry to the next which primitives may be added to the chain.

This function uses DMA to clear the ordering table. It prepares a reverse order list which starts at the last entry of the array and ends at the first. This is ideal for 3D graphics as higher table entries are drawn first and lower entries are drawn last. Primitives added to one entry first are always drawn last.

To begin processing of an ordering table array initialized by this function, execute DrawOTag(ot+n-1) (draw from last entry of array) since the ordering table is initialized with pointers in reverse order.

When adding an ordering table to another ordering table using addPrims, specify the last element for p0 and the first element for p1 if the ordering table is cleared by this function.

### See Also

**DrawOTag**

## DMACallback

Sets a callback routine for a DMA interrupt

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.10b* | *07/16/2019* |

### Syntax

| | |
|---|---|
| **void *DMACallback(** | |
| **int** *dma*, | DMA channel to set callback |
| **void** *(*func)()***)** | Callback function |

### Explanation

Sets a callback function specified by *func* to a DMA channel specified by *dma*, executed whenever a DMA transfer for the specified channel finishes. Calling this function will automatically install a handler on IRQ3 using **InterruptCallback** to handle DMA interrupts.

This function is not normally exposed to programmers in the official SDK, but is made available in LibPSn00b for low-level prototyping and advanced low-level programmers. Use this function only if you know exactly what you're doing.

The following lists the hardware device associated with each DMA channel:

| Channel | Device |
|---|---|
| 0 | MDEC input |
| 1 | MDEC output |
| 2 | GPU (used by libpsxgpu) |
| 3 | CD-ROM |
| 4 | SPU |
| 5 | PIO |
| 6 | OTC (used by libpsxgpu) |

Setting a DMA callback automatically adds an interrupt callback on IRQ3 using **InterruptCallback**(). If a callback routine on IRQ3 has already set, DMACallback will not set its own handler.

The callback is never an interrupt handler and a callback function must be written as a normal function. Since the callback function is called within an exception handler, function must return as soon as possible. Recursive function calls must be kept a minimum due to limited stack. DMA interrupt status bits are automatically acknowledged so the callback routine does not need to acknowledge it manually.

To uninstall a callback routine, simply specify NULL or 0 for *func*. It will also remove the IRQ enable bit of the corresponding DMA channel. If all DMA callbacks have been removed, the DMA callback handler is removed from the ISR.

### Returns

Pointer to the last installed callback routine.

## DrawOTag

Executes an ordering table

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *12/21/2018* |

### Syntax

**void DrawOTag(**
    **u_int** **ot***)**     Pointer to an ordering table to execute.

### Explanation

Executes primitives linked into an ordering table array specified by **ot*.

When executing an ordering table initialized by **ClearOTagR**, you must specify the last entry in the array.

DrawOTag uses DMA to query primitives to the GPU and may be non-blocking during DMA page gaps. Use **DrawSync** to ensure execution of the ordering table has completed.

### See also

**DrawSync ClearOTagR**

## DrawPrim

Draws a primitive

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *07/16/2019* |

### Syntax

**void DrawPrim(**
    **void** **pri***)**    Pointer to a primitives

### Explanation

Draws a primitive specified by *pri*. Uses software I/O to send the primitive to the GPU so its not recommended for drawing all graphics.

Use only for drawing a few primitives in a very simple single buffered menu for example.

## DrawSync

Waits until all GPU drawing or VRAM transfers have completed

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *12/21/2018* |

### Syntax

| | |
|---|---|
| **int DrawSync(** | |
| **int** *mode***)** | Function mode |

### Explanation

Waits until the GPU has finished processing drawing commands or VRAM transfers. If *mode* is non-zero, returns the number of words remaining in a DMA transfer.

### Work in progress

This function does not timeout if the GPU locks up due to a bad packet or corrupted ordering table as of version 0.09b.

### Returns

Number of words remaining in transfer if *mode* = 1.

## DrawSyncCallback

Sets a callback function that is executed on drawing or VRAM transfer completion

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.10b* | *07/17/2019* |

### Syntax

| | |
|---|---|
| **void *DrawSyncCallback(** | |
| **void** *(*func)()***)** | Pointer to a function |

### Explanation

Sets a callback function specified by *func* which will be executed on every drawing completion or VRAM transfer. Setting 0 will disable the callback.

Because the callback function is executed inside an interrupt handler, it is necessary to finish any processing as soon as possible. Sub function calls should be kept a minimum as the stack in the ISR is limited.

It is not recommended to issue VRAM or OT transfer operations within the callback function, use it only to set variables for keeping track of drawing and transfer completions.

It is recommended to define any variable manipulated by a callback function as **volatile**, to make sure any code reading the value will always receive changes.

### See also

**DrawSync**

## GetInterruptCallback

Returns the address of the callback function of a specified interrupt

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.10b* | *06/19/2019* |

### Syntax

**void *GetInterruptCallback(**
**int** *irq***)** Interrupt number

### Explanation

Gets the address of the callback function of an interrupt.

### Returns

Pointer to the callback function last set.

### See also

**InterruptCallback**

## GetTimInfo

Get image parameters of a TIM image file

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *Yes* | *0.01b* | *02/02/2019* |

### Syntax

| | |
|---|---|
| **int GetTimInfo(** | |
| **unsigned int** **tim,* | Pointer to a TIM image file |
| **TIM_IMAGE** **timimg***)** | Pointer to a TIM_IMAGE structure |

### Explanation

Retrieves parameters from a TIM file and stores relevant values to a **TIM_IMAGE** structure.

### Return value

0: success, 1: invalid file ID, 2: unsupported TIM version

### See also

**TIM_IMAGE**

## GetVideoMode

Gets the current video standard mode

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *2/2/2019* |

**Syntax**

**int GetVideoMode()**

**Explanation**

Returns the current video standard mode.

**Differences**

Unlike the official libraries, this function returns the video mode standard currently set (ie. If this function is called on a PAL machine while in a PAL display mode, it returns 1 or MODE_PAL).

**Returns**

MODE_NTSC: NTSC, MODE_PAL: PAL

**See also**

**SetVideoMode**

## InterruptCallback

Sets a callback routine for an interrupt

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.10b* | *07/16/2019* |

### Syntax

**void *InterruptCallback(**

| | |
|---|---|
| **int** *irq*, | Interrupt number to install callback |
| **void** *(*func)()***)** | Callback function |

### Explanation

Sets a callback function specified by *func* to the ISR, which is executed whenever an interrupt specified by *irq* occurs. Only one callback routine can be set per interrupt number at a time.

This is a special low-level function that is not normally used by programmers in the official SDK and is usually only called internally by the libraries. It is exposed in LibPSn00b for better control over the hardware in prototyping or when performing special operations. Use this function ONLY if you know exactly what you're doing.

The following lists the hardware device associated with each interrupt number:

| Interrupt | Device |
|---|---|
| 0 | Vsync (used by libpsxgpu) |
| 1 | GPU (triggered only by a special GPU packet) |
| 2 | CD-ROM |
| 3 | DMA |
| 4 | Timer 0 |
| 5 | Timer 1 |
| 6 | Timer 2 |
| 7 | Pad & Memory card |
| 8 | Serial (used by libpsxsio) |
| 9 | SPU |
| 10 | Light-gun & Expansion port |

Most hardware devices would only generate an interrupt when enabled by their I/O port registers.

This function should only be called while in critical section. The ISR automatically acknowledges interrupts so the callback routine does not need to acknowledge it (except hardware devices that additionally need to be acknowledged by their I/O registers). Avoid calling too many sub functions in the callback routine as the size of the stack in the ISR is limited.

To uninstall a callback routine, simply specify NULL or 0 for *func*. It will also remove the IRQ mask bit of the corresponding interrupt in the IRQ hardware registers which disables the interrupt.

### Returns

Pointer to the last installed callback routine.

## LoadImage

Upload image data to VRAM

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *12/21/2018* |

### Syntax

| | |
|---|---|
| **void LoadImage(** | |
| **RECT** **rect,* | Pointer to a RECT specifying VRAM destination coordinates |
| **unsigned int** **data***)** | Pointer to source image data |

### Explanation

Uploads image data from the source address *data* to VRAM. The image size and destination offset in VRAM is specified by *rect* using a RECT object.

LoadImage uses DMA to upload data to VRAM at high speed and could be non-blocking, use **DrawSync** to ensure the DMA transfer has completed. Using **DrawSync** when uploading multiple images at once is not necessary as LoadImage will wait for a previous transfer to complete.

If you want to upload a texture image on every frame in a real time sequence, it is best to perform the upload after a **DrawSync** call.

### See also

**DrawSync GetTimInfo**

## PutDrawEnv

Applies a DRAWENV structure

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.1b* | *06/07/2019* |

### Syntax

**void PutDrawEnv(**
    **DRAWENV** **draw***)** Pointer to a DRAWENV structures

### Explanation

Applies the specified **DRAWENV** structure to the GPU. This function is best called when the GPU is not busy processing any primitives. Use the DrawSync function to wait for the GPU to complete processing.

Alternatively, a DR_ENV struct can be used to change the drawing environment mid-drawing.

### See also

**DRAWENV**

## PutDispEnv

Applies a DISPENV structure

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

**Syntax**

| | |
|---|---|
| **void PutDispEnv(** | |
| **DISPENV** **disp***)** | Pointer to a DISPENV structure |

**Explanation**

Applies the specified **DISPENV** struct to the GPU. This function is best called immediately when a V-Blank occurs (using VSync) when updating the screen regularly. Use the VSync function to wait for a V-Blank to occur.

**See also**

**DISPENV** VSync

## ResetGraph

Resets the graphics subsystem

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**void ResetGraph(**
    **int** *mode***)**    Reset mode

### Explanation

Resets the GPU and graphics subsystem of libpsxgpu according to *mode*.

On first call, this function will additionally install its ISR subsystem to the kernel, installs its VSync callback, uninstall the BIOS CD subsystem and exit critical section regardless of *mode*. Because of this, it is highly recommended to call this function at the beginning of your program even if you don’t plan to do any graphics.

The following describes the behavior of the available mode numbers. The exact behavior in the official SDK is not known yet.

| Mode | Operation |
|---|---|
| 0 | Resets the GPU entirely including video mode (default of 256x240) and sets display mask to 0. |
| 1 | Cancels any DMA transfer and resets the GPU command buffer. |
| 3 | Resets the GPU command buffer. |

## SetDefDispEnv

Sets a display environment with default parameters

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Syntax

| | |
|---|---|
| **SetDefDispEnv(** | |
| **DISPENV** **disp,* | Pointer to a **DISPENV** structure |
| **int** *x,* **int** *y,* | X, Y framebuffer coordinates to display |
| **int** *w,* **int** *h***)** | Display resolution |

### Explanation

Prepares a **DISPENV** structure with the specified framebuffer and resolution coordinates using default video parameters.

The defaults are the *screen* element of **DISPENV** is set to zeroes, *isinter* is set 0 and *isrgb24* is set 0.

### See also

**DISPENV PutDispEnv**

## SetDefDrawEnv

Sets a drawing environment with default parameters

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Syntax

**SetDefDrawEnv(**

| | |
|---|---|
| **DRAWENV** **disp,* | Pointer to a **DRAWENV** structure |
| **int** *x,* **int** *y,* | X, Y framebuffer coordinates to draw to |
| **int** *w,* **int** *h***)** | Draw area size |

### Explanation

Prepares a **DRAWENV** structure with the specified framebuffer and resolution coordinates using default parameters.

The *ofs[]* elements of **DRAWENV** is set 0 (top-left), *tw* is set 0 (default texture window settings), *tpage* to 0x0a (640, 0), *dtd* to 1 (dithering enabled), *dfe* to 0 (don’t draw to displayed area), *isbg* to 0 (no draw area clear) and clear color values set to 0.

### See also

**DRAWENV PutDrawEnv**

## SetDispMask

Sets the display mask

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/08/2019* |

### Syntax

| | |
|---|---|
| **void SetDispMask(** | |
| **int** *mask***)** | Display mask setting (0: no display, 1: display) |

### Explanation

Sets the display mask of the GPU. If *mask* is 0, the console will only show a black screen.

This function is useful for hiding garbage shown during video init/setup. **ResetGraph** automatically sets the display mask to 0.

Best called after **VSync** and **PutDispEnv**.

## SetVideoMode

Sets the video standard

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/17/2019* |

### Syntax

**void SetVideoMode(**
    **int** *mode***)**    Video standard to set

### Explanation

Sets the video standard by *mode* (MODE_NTSC for NTSC or MODE_PAL for PAL), normally used to override the video standard of the console.

Keep in mind that using a video standard other than the one specified by the region of the console will result to color problems or unstable picture without modifications, depending on the model of the console.

## StoreImage

Download image data from VRAM

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *07/16/2019* |

### Syntax

| | |
|---|---|
| **void StoreImage(** | |
| **RECT** **rect,* | Pointer to a RECT specifying VRAM source coordinates |
| **unsigned int** **data***)** | Pointer to store downloaded image data |

### Explanation

Downloads a portion of VRAM from an area specified by *rect,* and stores the downloaded pixel data to a buffer specified by *data*.

StoreImage uses DMA to upload data to VRAM at high speed and could be non-blocking, use **DrawSync** to ensure the DMA transfer has completed.

### See also

**DrawSync**

## VSync

Wait for vertical retrace, or return elapsed vertical blank counter, or hblank count since last call

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/23/2019* |

### Syntax

**void VSync(**
    **int** *mode***)**    Mode

### Explanation

Waits until a vertical retrace occurs or returns a value using the method specified by *mode*, as defined below.

| Mode | Operation |
|---|---|
| 0 | Waits until a vertical retrace event occurs. |
| 1 | Only return the Hblank count elapsed since last VSync call. |
| n>1 | Waits until n vertical retrace events occur. |
| n<0 | Returns number of vertical retrace events elapsed since the start of the program. |

VSync() will timeout if the vertical blanking interrupt stops working either due to calling ChangeClearPAD(1), or calling _InitPad() without calling ChangeClearPAD(0) next. The function will attempt to restart vertical blanking interrupts by calling ChangeClearPAD(0) and ChangeClearRCnt(3, 0).

VSync() may also timeout if a large wait value is specified. Use a for-loop that calls VSync(0) instead to get around this limitation.

### Return value

Return value varies depending on the value specified by *mode*.

| Mode | Return value |
|---|---|
| >=0 | Hblank count elapsed since last VSync call. |
| <0 | Number of vertical retrace events elapsed since the start of your program. |

### See also

**VSyncCallback**

## VSyncCallback

Sets a specified function to be executed on every V-blank

| Library | Header | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *liblibpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/17/2019* |

### Syntax

**void *VsyncCallback(**
    **void** *(*func)()***)** Pointer to a callback function

### Explanation

Sets a callback function specified by *func* called on every V-blank. Setting 0 will disable the callback.

Because the callback function is executed during a critical section inside an ISR, it is necessary to finish any processing quickly. Sub function calls should also be kept at minimum as the stack in the ISR is limited.

It is recommended to define any variable manipulated by a callback function as **volatile** to make sure that any loop reading the value will always read the variable for changes.

### Returns

Pointer to last callback function set.

### See also

**VSync**

# Macros

## addPrim

Links a primitive packet to an ordering table

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

| | |
|---|---|
| **addPrim(** | |
| *ot,* | Pointer to an ordering table element |
| *p***)** | Pointer to a primitive packet |

### Explanation

Links a primitive packet to an ordering table element by setting the value from the specified table element to the primitive packet's tag element (with the size byte retained) and the pointer to the packet is set to the specified table element.

It is recommended to generate primitive packets in a global buffer to ensure that they do not get overwritten when the GPU gets around to processing the primitive (ie. If you allocate the primitive as a local variable in a function, it may have been overwritten when the GPU gets to draw it).

A common misconception among PS1 homebrew programmers is that they sometimes believe that only a single primitive packet can only be added to each ordering table element. This is false as adding another primitive to an ordering table element that already has a primitive added to it will only add to the chain, not replace it so pretty much any number of primitives can be added to a single table element. Therefore, an ordering table length of 4 to 8 elements is usually enough for a 2D game project.

### See also

**ClearOTagR DrawOTag**

## addPrims

Links an ordering table to another ordering table

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**addPrims(**

| | |
|---|---|
| *ot,* | Pointer to an ordering table element |
| *p0,* | Pointer to the first element of the ordering table to add |
| *p1***)** | Pointer to the last element of the ordering table to addition |

### Explanation

This macro links one ordering table specified by *p0* and *p1* to another ordering table.

The ordering table element that is considered the first element in the chain depends on which function was used to prepare the ordering table. If the ordering table was cleared using ClearOTagR the last element of the array is the first and the first element is the last, if the ordering table is cleared using ClearOTag the first element in the array is the first and the last element is the last.

### See also

**ClearOTagR**

## getClut

Calculates and returns a CLUT value

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/17/2019* |

### Syntax

**getClut(**

| | |
|---|---|
| *x, y***)** | Framebuffer coordinates to a CLUT |

### Explanation

Calculates a CLUT value from the specified coordinates. The resulting value is used on textured primitives with a CLUT field. *x* must be a multiple of 16 units, the value will be rounded down to the nearest lower multiple otherwise.

A CLUT is needed only if the texture color depth is 4-bit or 8-bit.

Primitives with a CLUT field include **SPRT**, **SPRT_8, SPRT_16**, **POLY_FT3, POLY_FT4** and **POLY_GT3, POLY_GT4**.

## getTPage

Calculates and returns a texture page value

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/16/2019* |

### Syntax

**getTPage(**

| | |
|---|---|
| *tp,* | Texture color depth (0: 4-bit, 1: 8-bit, 2: 16-bit) |
| *abr,* | Blend operator mode (see below) |
| *x, y***)** | Framebuffer coordinate of texture page |

### Explanation

Calculates a texture page value using the specified coordinates. The resulting value is used with textured primitives that have a Tpage field or a DR_TPAGE primitive (using **setDrawTPageVal**).

The framebuffer coordinates should be a multiple of 64 for the X axis and a multiple of 256 for the Y axis, the coordinates will be rounded down to the nearest lower multiple otherwise.

**The following lists the blend modes for semi-transparent primitives (*abr*):**

| Mode | Operation | | | |
|---|---|---|---|---|
| 0 | B:50% | + | F:50% | (50% alpha) |
| 1 | B:100% | + | F:100% | (additive) |
| 2 | B:100% | - | F:100% | (subtractive) |
| 3 | B:100% | - | F:25% | (subtract 25%) |

Primitives that have a Tpage field include **POLY_FT3, POLY_FT4** and **POLY_GT3, POLY_GT4**, use **DR_TPAGE** and **setDrawTPage** or **setDrawTPageVal** for textured primitives without a Tpage field.

### Returns

16-bit texture page value.

### See also

**setDrawTPage setDrawTPageVal**

## setClut

Sets the CLUT field of a primitive by coordinates

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/17/2019* |

### Syntax

**setClut(**

| | |
|---|---|
| *p,* | Pointer to a primitive struct with a CLUT field |
| *x, y***)** | Framebuffer coordinates to a CLUT |

### Explanation

Sets the CLUT field of a primitive by framebuffer coordinates. *x* must be a multiple of 16 pixels, the value will be rounded down to the nearest lower multiple otherwise.

Primitives with a CLUT field include **SPRT**, **SPRT_8, SPRT_16**, **POLY_FT3, POLY_FT4** and **POLY_GT3, POLY_GT4**.

### See also

**getClut**

## setDrawMask

Prepares a DR_MASK primitive

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *Yes* | *0.01b* | *07/17/2019* |

### Syntax

**setDrawMask(**

| | |
|---|---|
| p, | Pointer to a DR_MASK primitive |
| sb, | Set mask bit on pixels drawn (0: don't set, 1: set) |
| mt**)** | Mask test (0: draw always, 1: don't draw on masked pixels) |

### Explanation

Prepares and sets the specified values to a DR_MASK primitive. The mask feature allows for limited stencil effects with the GPU.

Setting *sb* to 1 makes primitives set the mask bit on every pixel drawn, the mask bit is stored on the 16th bit of each pixel within the drawing area. The mask is cleared by primitives if *sb* is set 0.

Textured primitives with semi-transparency bits set on either the pixels or CLUT colors of the texture will also set this mask bit regardless of the *sb* setting. Setting *mt* to 1 enables mask test, which prohibits drawing on areas that have the mask bit set in the drawing area.

The mask settings affects all GPU drawing packets as well as GPU VRAM transfer and move operations, it is recommended to issue a DR_MASK with *sb:0* and *mt:0* to reset the mask settings after your mask effects.

### See also

**DR_MASK**

## setDrawTPage

Prepares a DR_TPAGE primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/16/2019* |

### Syntax

**setDrawTPage(**

| | |
|---|---|
| *p,* | Pointer to a DR_TPAGE primitive |
| *tp,* | Texture color depth (0: 4-bit, 1: 8-bit, 2: 16-bit) |
| *abr,* | Blend operator mode (see **getTPage**) |
| *x, y***)** | Framebuffer coordinate of texture page |

### Explanation

Prepares and sets the specified values to a DR_TPAGE primitive, used to change the current Tpage of the GPU mid-drawing for primitives that do not have a Tpage field, and/or to set a blending operator for semi-transparent, non-textured primitives.

The framebuffer coordinates should usually be a multiple of 64 for the X axis and a multiple of 256 for the Y axis, the coordinates will be rounded down to the nearest lower value otherwise. Texture color depth has no effect on framebuffer coordinates.

### See also

**DR_TPAGE**

## setDrawTPageVal

Prepares a DR_TPAGE primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *Yes* | *0.01b* | *07/16/2019* |

### Syntax

**setDrawTPageVal(**

| | |
|---|---|
| *p*, | Pointer to a DR_TPAGE primitive |
| *tp***)** | Tpage value to set (see **getTPage**) |

### Explanation

Prepares a DR_TPAGE primitive similar to **setDrawTPage**, but takes a single Tpage value rather than a bunch of parameters that define a Tpage.

A Tpage value is normally acquired using **getTPage**.

### See also

**DR_TPAGE**

## setLineF2

Prepares a LINE_F2 primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *07/16/2019* |

### Syntax

**setLineF2(**

| | |
|---|---|
| *p***)** | Pointer to a LINE_F2 primitive |

### Explanation

Prepares a LINE_F2 packet by setting the appropriate packet size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**LINE_F2, LINE_F3, LINE_F4**

## setLineF3

Prepares a LINE_F3 primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *07/16/2019* |

### Syntax

| | |
|---|---|
| **setLineF3(** | |
| *p***)** | Pointer to a LINE_F3 primitive |

### Explanation

Prepares a LINE_F4 packet by setting the appropriate packet size and code values to the primitive, and sets a terminator word at the end of the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**LINE_F2, LINE_F3, LINE_F4**

## setLineF4

Prepares a LINE_F4 primitives

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

**Syntax**

**setLineF4(**
*p***)** Pointer to a LINE_F4 primitive

**Explanation**

Prepares a LINE_F4 packet by setting the appropriate packet size and code values to the primitive, and adds a terminator word at the end of the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

**See also**

**LINE_F2, LINE_F3, LINE_F4**

## setLineG2

Prepares a LINE_G2 primitive

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *06/07/2019* |

### Syntax

**setLineG2(**
*p***)** Pointer to a LINE_G2 primitive

### Explanation

Prepares a LINE_G2 packet by setting the appropriate size and code values to the primitive, and adds a terminator word at the end of the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**LINE_G2, LINE_G3, LINE_G4**

## setLineG3

Prepares a LINE_G3 primitive

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.12b* | *06/07/2019* |

### Syntax

**setLineG3(**
*p***)** Pointer to a LINE_G3 primitive

### Explanation

Prepares a LINE_G3 packet by setting the appropriate size and code values to the primitive, and adds a terminator word at the end of the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**LINE_G2, LINE_G3, LINE_G4**

## setLineG4

Prepares a LINE_G4 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/16/2019* |

### Syntax

**setLineG4(**
*p***)** Pointer to a LINE_G4 primitive

### Explanation

Prepares a LINE_G4 packet by setting the appropriate size and code values to the primitive, and adds a terminator word at the end of the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**LINE_G2, LINE_G3, LINE_G4**

## setPolyF3

Prepares a POLY_F3 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

**Syntax**

**setPolyF3(**

| | |
|---|---|
| *p***)** | Pointer to a **POLY_F3** primitive |

**Explanation**

Prepares a **POLY_F3** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

**See also**

**POLY_F3**

## setPolyFT3

Prepares a POLY_FT3 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setPolyFT3(**
*p***)** Pointer to a **POLY_FT3** packet

### Explanation

Prepares a **POLY_FT3** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates, tpage, clut and color) to the primitive and before adding it to an ordering table using addPrim.

### See also

**POLY_FT3**

## setPolyG3

Prepares a POLY_G3 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setPolyG3(**

| | |
|---|---|
| *p***)** | Pointer to a **POLY_G3** packet |

### Explanation

Prepares a **POLY_G3** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**POLY_G3**

## setPolyGT3

Prepares a POLY_GT3 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setPolyGT3(**

| | |
|---|---|
| *p***)** | Pointer to a POLY_G3 packet |

### Explanation

Prepares a POLY_GT3 packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates, tpage, clut and color) to the primitive and before adding it to an ordering table using addPrim.

### See also

POLY_GT3

## setPolyF4

Prepares a POLY_F4 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setPolyF4(**

| | |
|---|---|
| *p*) | Pointer to a **POLY_F4** packet |

### Explanation

Prepares a **POLY_F4** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**POLY_F4**

## setPolyFT4

Prepares a POLY_FT4 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setPolyFT4(**

| | |
|---|---|
| *p***)** | Pointer to a **POLY_FT4** packet |

### Explanation

Prepares a **POLY_FT4** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates, tpage, clut and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**POLY_FT4**

## setPolyG4

Prepares a POLY_G4 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

**Syntax**

**setPolyG4(**

| | |
|---|---|
| *p***)** | Pointer to a **POLY_G4** packet |

**Explanation**

Prepares a **POLY_G4** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

**See also**

**POLY_G4**

## setPolyGT4

Prepares a POLY_GT4 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setPolyGT4(**

| | |
|---|---|
| *p***)** | Pointer to a POLY_GT4 packet |

### Explanation

Prepares a POLY_GT4 packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates, tpage, clut and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

POLY_GT4

## setRECT

Sets coordinates to a RECT struct

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/17/2019* |

### Syntax

**setRECT(**

| | |
|---|---|
| *v,* | Pointer to a RECT struct |
| *_x,* | X coordinate to set |
| *_y,* | Y coordinate to set |
| *_w,* | Width coordinate to set |
| *_h***)** | Height coordinate to set |

### Explanation

Sets the x, y, w, and h fields of a RECT specified by *v*, with coordinates specified by *_x, _y,* _w and *_h*. Cleaner looking to use over setting the fields directly.

### See also

**RECT**

## setSprt

Prepares a SPRT primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setSprt(**
*p***)** Pointer to a **SPRT** packet

### Explanation

Prepares a **SPRT** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y,  coordinates, clut and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**SPRT**

## setSprt8

Prepares a SPRT_8 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setSprt8(**
*p***)** Pointer to a **SPRT_8** packet

### Explanation

Prepares a **SPRT_8** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates, clut and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**SPRT_8**

## setSprt16

Prepares a SPRT primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setSprt16(**
*p***)** Pointer to a **SPRT_16** packet

### Explanation

Prepares a **SPRT_16** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates, clut and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**SPRT_16**

## setTexWindow

Prepares a DR_TWIN primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *none* | *psxgpu.h* | *No* | *0.14b* | *10/22/2019* |

### Syntax

**setTexWindow(**

| | |
|---|---|
| *p,* | Pointer to a DR_TWIN structure |
| *r***)** | Pointer to a **RECT** structure |

### Explanation

Prepares a DR_TWIN primitive by setting the packet size and packet code based on arguments specified.

The (x, y) coordinates in the RECT structure specifies the offset of the texture window in units of 8 pixels (1 = 8 pixels). The offset adds to the (u,v) coordinates of any textured primitive.

The (w, h) coordinates specifies the texture window constraint in units of 8 pixels (1 = 8 pixels). The constraint limits the range of pixels that can be read, and wraps pixels when texture coordinates exceed the size of the constraint.

## setTile

Prepares a TILE primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setTile(**
*p***)** Pointer to a **TILE** packet

### Explanation

Prepares a **TILE** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**TILE**

## setTile1

Prepares a TILE_1 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setTile(**
*p***)** Pointer to a **TILE_1** packet

### Explanation

Prepares a **TILE_1** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**TILE_1**

## setTile8

Prepares a TILE_8 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

**Syntax**

**setTile8(**
*p***)** Pointer to a **TILE_8** packet

**Explanation**

Prepares a **TILE_8** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

**See also**

**TILE_8**

## setTile16

Prepares a TILE_16 primitive

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *06/07/2019* |

### Syntax

**setTile16(**

| | |
|---|---|
| *p***)** | Pointer to a **TILE_16** packet |

### Explanation

Prepares a **TILE_16** packet by setting the appropriate size and code values to the primitive.

Use this macro before setting other values (x,y coordinates and color) to the primitive and before adding it to an ordering table using **addPrim**.

### See also

**TILE_16**

## setTPage

Sets the Tpage of a primitive by coordinates

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/17/2019* |

### Syntax

**setTPage(**

| | |
|---|---|
| *p,* | Pointer to a primitive with a Tpage field |
| *tp,* | Texture color depth (0: 4-bit, 1: 8-bit, 2: 16-bit) |
| *abr,* | Semi-transparency blend operator (see **getTPage**) |
| *x, y***)** | Framebuffer coordinates to a texture page |

### Explanation

Sets the Tpage field of a primitive by coordinates.

Primitives that have a Tpage field include **POLY_FT3, POLY_FT4** and **POLY_GT3, POLY_GT4**.

### See also

**getTPage**

## setVector

Sets coordinates to a VECTOR or SVECTOR struct

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxgpu.a* | *psxgpu.h* | *No* | *0.01b* | *07/17/2019* |

### Syntax

| | |
|---|---|
| **setVector(** | |
| *v,* | Pointer to a VECTOR or SVECTOR struct |
| *_x,* | X coordinate to set |
| *_y,* | Y coordinate to set |
| *_z***)** | Z coordinate to set |

### Explanation

Sets the vx, vy and vz fields of a VECTOR or SVECTOR struct specified by *v*, with coordinates specified by *_vx, _vy* and *_vz*. Cleaner looking to use over setting the fields directly.

# Miscellaneous Library

## Chapter Contents

## Overview

The miscellaneous library provides functions mostly to aid in prototyping and testing.

## Functions

### FntLoad

Upload debug font texture to VRAM

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxetc.a* | *psxetc.h* | *No* | *R1* | *09/25/2019* |

**Syntax**

| | |
|---|---|
| **void FntLoad(** | |
| **int** *x*, **int** *y***)** | Framebuffer coordinates to upload font texture |

**Explanation**

Uploads the font texture to VRAM, so debug text drawing functions can be used. This function must be called first before using **FntOpen()**, **FntPrint()** and **FntFlush()**.

The size of the font texture is 32x64 plus a 16 color CLUT immediately below the texture. The X coordinate must be a multiple of 64 and the Y coordinate a multiple of 256.

This function can also close all text streams previously created by **FntOpen()**.

**See also**

**FntOpen**

## FntOpen

Opens a debug font text stream

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxetc.a* | *psxetc.h* | *No* | *0.13b* | *09/25/2019* |

### Syntax

| | |
|---|---|
| **int FntOpen(** | |
| **int** *x*, **int** *y*, | X,Y coordinate of text window |
| **int** *w*, **int** *h*, | Width and height of text window |
| **int** *isbg*, | Draw background (0: none, 1: black, 2: semi-transparent black) |
| **int** *n***)** | Number of characters to allocate |

### Explanation

Opens a text stream window using the debug font uploaded by FntLoad().

The text will only draw inside the area specified by (*x*,*y*)-(*w*,*h*), to allow you to crete multiple text streams at different portions of the screen. The text will wrap if it passes the size of the specified window area. The coordinates are draw area relative and not framebuffer absolute, so you don’t have to adjust it relative to your current draw area coordinates.

*Isbg* specifies if a solid background should be drawn below the text to improve text readability. Specifying 1 draws a solid black rectangle as the text background, while a value of 2 draws a semi-transparent black rectangle, which not only improves text readability but also allow graphics behind the window to be visible.

*n* specifies how many characters to allocate for the text stream.

Up to 8 text streams can be created at once. Previously opened streams can be closed and deallocated using **FntLoad**.

### Returns

Number of text stream opened, -1 if no more streams can be opened.

### See also

**FntLoad FntPrint**

## FntPrint

Print text to specified text stream

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxetc.a* | *psxetc.h* | *No* | *0.13b* | *09/25/2019* |

### Syntax

| | |
|---|---|
| **int FntPrint(** | |
| **int** *id*, | Stream number (-1 = use last opened stream) |
| **const char** **fmt*, | Format string (same syntax as printf()) |
| … **)** | Text format arguments |

### Explanation

Prints text to the specified text stream created by **FntOpen**.

This function works much like fprintf(), but text output is directed to the debug font text stream. *Id* specifies which text stream created by **FntOpen** to print the text to, or specify -1 to write the text to the last opened stream.

Because of modern GCC requiring at least one named argument in function names, this function does not have the same syntax as **FntPrint** in the official SDK, and a stream number must be specified at all times.

Use **FntFlush** to draw the text written in the specified text stream.

### Returns

Number of characters written.

### See also

**FntLoad FntOpen FntFlush**

## FntFlush

Draws a text stream

| Library | Header File | Original | Introduced | Documentation Date |
|---|---|---|---|---|
| *libpsxetc.a* | *psxetc.h* | *No* | *0.13b* | *09/25/2019* |

### Syntax

**char *FntFlush(**
    **int** *id***)**    Stream number (-1 = use last opened stream)

### Explanation

Draws the text window and characters of the specified text stream.

The function waits for drawing to complete, then draws the primitives using DMA transfer and finally waits for it to complete. This helps ensure the text primitives are fully drawn, though it may result to some performance loss.

### Returns

Pointer to an internal primitive buffer used to draw the text stream, can be drawn using **DrawOTag**.

### See also

**FntLoad FntOpen FntPrint**

# Serial Input/Output Library

## Chapter Contents

## Overview

The serial library provides functions to configure and control the serial interface of the PSX. It also provides a custom device intended to replace the default tty device to direct tty output from printf() calls to the serial interface, to be viewed in a serial terminal.

The PSX's serial interface is capable of baud rates of up to 1Mbaud but 230400 baud is the highest data rate that USB serial adapters (such as a CH340) can receive. Achieving reliable communications with high data rates is yet to be studied further.

# Functions

## _sio_control

Serial control function

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxsio.a* | *psxsio.h* | *No* | *0.10b* | *07/16/2019* |

### Syntax

| | |
|---|---|
| **int _sio_control(** | |
| **int** *cmd,* | Command |
| **int** *arg,* | Subcommand |
| **int** *param***)** | Parameter |

### Explanation

Multi-purpose serial control function, used to control and retrieve every aspect of the serial interface.

The behavior of this function varies depending on the values specified by *cmd* and *arg.*

**The following describes command value combinations:**

| cmd | arg | Function |
|---|---|---|
| 0 | 0 | Read serial status register. |
| 0 | 1 | Read serial control register. |
| 0 | 2 | Read serial mode register. |
| 0 | 3 | Read serial baud rate. |
| 0 | 4 | Read 1 byte from serial interface (returns byte received). |
| 1 | 1 | Set serial control register. |
| 1 | 2 | Set serial mode (parameters specified by *param*). |
| 1 | 3 | Set serial baud rate (value specified by *param*). |
| 1 | 4 | Write 1 byte to serial interface (byte value specified by *param*). |
| 2 | 0 | Reset serial interface. |
| 2 | 1 | Acknowledge serial interrupt and comms errors. |

**The following describes serial control options (some values not documented in official SDK):**

| Bits | Definition | Description |
|---|---|---|
| 0 | CR_TXEN | TX enable. |
| 1 | CR_DTR | Output DTR signal. |
| 2 | CR_RXEN | RX enable. |
| 3 | CR_BRK | Invert TX logic levels. |
| 4 | CR_INTRST | Acknowledge IRQ and comms errors. |
| 5 | CR_RTS | Output RTS signal. |
| 6 | CR_ERRRST | Reset serial hardware. |
| 7 | | Unknown (always 0). |
| 8-9 | | Interrupt when RX buffer has n bytes. |
| | CR_BUFSIZ_1 | 00: Interrupt on 1 byte. |
| | CR_BUFSIZ_2 | 01: Interrupt on 2 bytes. |
| | CR_BUFSIZ_4 | 10: Interrupt on 4 bytes. |
| | CR_BUFSIZ_8 | 11: Interrupt on 8 bytes. |
| 10 | CR_TXIEN | Interrupt on TX ready. |
| 11 | CR_RXIEN | Interrupt on RX receive. |
| 12 | CR_DSRIEN | Interrupt on DSR signal. |
| 13-15 | | Unused (always zero). |

**The following describes serial mode options:**

| Bits | Definition | Description |
|---|---|---|
| 0-1 | None | Baud rate reload factor (must be 0x2 always). |
| 2-3 | | Character length. |
| | MR_CHLEN_5 | 00: 5 bits per word. |
| | MR_CHLEN_6 | 01: 6 bits per word. |
| | MR_CHLEN_7 | 10: 7 bits per word. |
| | MR_CHLEN_8 | 11: 8 bits per word. |
| 4 | MR_PEN | Parity enable. |
| 5 | MR_P_EVEN | Odd parity (definition is misleading). |
| 6-7 | | Stop bit length. |
| | MR_SB_01 | 01: 1 stop bit. |
| | MR_SB_10 | 10: 1.5 stop bits. |
| | MR_SB_11 | 11: 2 stop bits. |
| 8-15 | | Unused (always zero). |

**The following describes serial status bits:**

| Bits | Definition | Description |
|---|---|---|
| 0 | SR_TXRDY | TX ready. |
| 1 | SR_RXRDY | Bytes pending in RX buffer. |
| 2 | SR_TXU | TX completed. |
| 3 | SR_PERROR | Parity error. |
| 4 | SR_OE | RX buffer overflow. |
| 5 | SE_FE | RX bad stop bit. |
| 6 | | RX input level. |
| 7 | SR_DSR | DSR signal level. |
| 8 | SR_CTS | CTS signal level. |
| 9 | SR_IRQ | Interrupt request. |
| 10 | | Unknown (always zero). |
| 11-25 | | 15-bit baud rate timer. |

## AddSIO

Installs a serial tty device

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxsio.a* | *psxsio.h* | *No* | *0.10b* | *06/14/2019* |

### Syntax

**void AddSIO(**
  **int** *baud***)** Baud rate.

### Explanation

Replaces the default BIOS tty device (and Caetla's tty device) with a serial tty device which redirects all stdout output (such as printf) to serial. The data rate is specified by *baud*, the rest of the parameters are 8 data bits, 1 stop bit, no parity and no hardware handshake by default.

This function can be called at the very beginning of your program (even before **ResetGraph**) to receive every printf message in your program.

## DelSIO

Deletes the serial tty device

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxsio.a* | *psxsio.h* | *No* | *0.10b* | *06/14/2019* |

**Syntax**

**void DelSIO(void)**

**Explanation**

Deletes the serial tty device, not recommended as any further tty output will likely crash the system.

## WaitSIO

Waits for serial

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxsio.a* | *psxsio.h* | *Yes* | *0.10b* | *06/14/2019* |

**Syntax**

**void WaitSIO(void)**

**Explanation**

Waits until a single byte is received from the serial interface, intended to be called immediately after AddSIO and is useful for pausing your program so you can open a terminal program and receive all tty messages.

## Sio1Callback

Sets a serial callback routine

| Library | Header File | Original | Introduced | Date Documented |
|---|---|---|---|---|
| *libpsxsio.a* | *psxsio.h* | *No* | *0.10b* | *06/14/2019* |

**Syntax**

| | |
|---|---|
| **void *Sio1Callback(** | |
| **void (**func*)(void))** | Callback function. |

**Explanation**

Sets a function specified by *func* as a callback routine that is executed whenever the serial interface generates an interrupt enabled by CR_TXIEN, CR_RXIEN or CR_DSRIEN using **_sio_control**(1, 1, <param>). If *func* is zero, the callback is disabled.

It is recommended to read at least 1 byte from the serial interface and call **_sio_control**(2, 1, 0) to acknowledge the serial interrupt at the end of your callback routine.

Since the callback function is executed in the global ISR, sub function calls must be kept at minimum due to limited stack available. The callback function must return as soon as possible to avoid missing any further interrupt requests.

**Return value**

Address of previously set callback function.