

---

[ [Welcome](#) | [What's New?](#) | [Installation](#) | [Technical Support](#) ]  
[ [Overview of Tools on CD](#) | [Index of Tools on CD](#) | [Samples](#) ]

---



This CD contains over 140 samples. The majority were written by Sony Computer Entertainment (Japan), and demonstrate officially accepted methods of calling program API's and initializing structures. If you are curious about the "garbage" text in the comments of the sample code, don't worry: these are Japanese comments that have a corresponding English translation provided beside it. The remaining samples were contributed by members of SCEE, SCEA, and generous developers. If **you** would like to donate some code for inclusion on the next Programmer Tools CD, refer to the section below entitled [Can I contribute my own samples?](#).

Many of the samples can be run directly from the Programmer Tools CD-ROM from your development system's CD-ROM drive; read the section below on [Running the Samples from the CD](#).

## What Samples are Available?

A text only listing of the samples, along with a brief description, appears [\psx\simplst.txt](#). Note that the text only lists samples which are officially provided by SCE Japan.

More samples, contributed by members of SCEA and SCEE, are listed in the document you are currently reading. In addition, we've added some caveats, warnings, and hyperlinks. Have a look at the categories below:

- [CD related samples](#) Examples that demonstrate how to stream the audio and video data from the movies and audio tracks on the Programmer Tools CD.
- [Scratch pad sample](#). A small sample showing how to access the D-Cache, which is an extremely fast 1 Kbyte cache.
- [Thread sample](#). A small sample illustrating how to use threads.
- [Controller samples](#). Code shows you how to interface with the analog controller, light gun, mouse, multitap, and the serial link cable (combat cable).
- [Kanji samples](#). If you're developing for the Japanese market, you will find the code here handy.
- [Sound samples](#). XA audio, streaming, MIDI sequenced files.
- [Graphics Samples](#). The whole spectrum of manipulating 2D and 3D graphics is presented here: HMD **NEW**, sprites, manipulating 3D modeling data (in PlayStation TMD format), drawing polygons on the screen, and moving your virtual camera around.

- [Memory Card Samples](#). Don't forget about these samples for the memory card; they illustrate how to read and write to the card, and even make the card act like a security dongle.
- [Module and menu samples](#). These samples demonstrate how PlayStation programs can launch other PlayStation programs. If you need to have a menu launcher, or you want to partition your game in modules, use the examples here. These show you how to run overlays, or execute programs using "LoadExec( )".
- [MPEG decompression](#). These samples demonstrate which parameters you need to toggle in order to make your MPEG compressed video looks its best.
- [SCEA samples](#). Miscellaneous contributed samples obtained from SCEA and North American developers. Other contributed sample are listed in the relevant categories. Contains an example of using the PC file system.
- [SCEE samples](#). Miscellaneous contributed samples obtained from SCEE and European developers. Other contributed sample are listed in the relevant categories. Contains the European demo disc, a profiler, a PAL example, and others.
- [Serial Cable samples](#). On the blue debugging PlayStation, a serial port can be used to communicate with the serial port of an IBM PC-AT compatible, via the DTL-S3050 cable.

This huge number of samples is overwhelming, especially if you are new to PlayStation programming. I would suggest that you start with the "tuto" directories of the categories, if they exist. They contain basic programs, which are nevertheless pretty advanced for a newbie. Use the debugger "dbugpsx.exe" to step through the code.

In order to learn anything, you really need to install the (included) Adobe Acrobat Reader 3.0 (or later) and have the Technical Reference CDROM sitting in your CD-ROM drive. You will **constantly** need to use the Reader's searching capabilities to help track down the function definitions (in the Library Reference manual) and the general concepts (in the Library Overview manual). **For a simple tutorial on using Adobe Acrobat**, click [here](#).

## About the contributed samples

A note about the contributed samples. While the samples have been attributed to various authors, it's well known that authors borrow code from other programmers too. So many names are left unlisted in this document, but we salute you all. In addition, some of the older samples may not work for you, being based on old libraries, or on data files that we don't have. Therefore, please use them at your own risk, and contact the authors if you have problems.

## Can I contribute my own samples ?

Yes, please send them to us! Refer to the document [\psx\sample\contrib\readme.txt](#) for information on contributing your samples. That document also refers to [\psx\sample\contrib\agree.pdf](#) which will be worked out with your publisher's lawyers to allow your source code to be posted (**you** don't have to deal with this document yourself, but it's provided for your curiosity. Please note that this document is needed to enable us to publish your code without suffering from accusations that we've **stolen** the code unlawfully!)

## To Install the Samples:

- If you're running under Windows 95, launch the "setup.exe" available on this CDROM.
- Or, simply copy the samples from the Programmer Tools CD into your root PlayStation

development directory, such as "c:\ps\psx\samples". Make sure you maintain the the directory structure of the Programmer Tools CDROM. In the future, we may add HTML readme's to the contents of the directories themselves, and this will rely on hard-coded relative path-names.

## To Run the Samples:

Note that most of the samples are precompiled. To view the samples, you have two options:

- **Running (almost) all of the samples all at once.** If you just want to run the samples on your development board, there is a program in \psx\sample\module\menu that allows you to do this. For a simple set of instructions read the section below on [Running the Samples from the CD](#).
- **Running samples individually.** For a discussion on how to compile and run the samples, click [here](#).

## What's changed?

The official text-only listing of the changes is in the document [\psx\smgchg.txt](#). In addition, this document contains a full listing of the samples. Items tagged with **NEW** are new to this release relative to the previous release of the programmer tools CD.

---

## Running the Samples from the CD

In the past, developers have wanted to run all of the samples directly from the Programmer Tools CD. It's possible to do something like that by using the CD in conjunction with your development boards, their associated CD-ROM drives (DTL-H2010 or DTL-H2510), and your PC's hard drive. Rather than running the samples from your CD-ROM, you copy the .exe files (in the procedure listed below) to your hard-drive, and run them from there. Here's what you need to do:

1. Make sure your development system is up and can run at least one of the samples.
2. Make sure the CD-ROM drive is hooked up to your development board.
3. Copy the directory [\psx](#) directly from the CD-ROM to the root directory of your hard-drive. It's very important that you place this in the root directory, since a script expects it to be there. If you are low on drive space, you can remove the "lib" and "include" directories. The whole "psx" hierarchy consumes about 90 Megabytes.
4. Put the Programmer Tools CD into the CD-ROM drive (the DTL-H2010 or the DTL-H2510, *not your computer's CD-ROM drive*). Some of the sample programs will access the CD-DA tracks, the XA tracks, or the movie streaming data (in the \Xdata directory).
5. Tell the development board to select the CD-ROM drive by typing the following command (assuming that you set up your binaries in c:\ps\pssn\bin): "run c:\ps\pssn\bin\selcd.cpe". (If you don't know where your "selcd.exe" file is, run the "find" utility under Windows 95, or change to the root directory of your hard drive and type "dir /s /b selcd.cpe" at an MS-DOS prompt.)
6. Change to the directory \psx\sample\module\menu: type "cd c:\psx\sample\module\menu".
7. Reset the development board: "resets 1", and (if you're running on a DTL-H2000), "run c:\ps\pssn\bin\snpatch.cpe".
8. Run the program "pcmenu.cpe" by typing the command "run pcmenu.cpe" at an MS-DOS prompt. If everything's working correctly, you should see a colorful screen.

9. Run the program "mess1.com" (in the same MS-DOS command window) so that you can see the printed message. Then type "testmess".
10. Use your controller's up and down pads to navigate through the directories. You may see a Japanese description of the program; hit the "SELECT" button to switch to the English translation. To run a program, hit the "START" button, and the child program will start. To exit the child program, hit the "SELECT" and "START" buttons simultaneously. (Unfortunately, not all of the samples follow this exiting rule, and are so noted in the descriptions below. In addition, you can simply read the "readme.txt" in the particular sample's directory to get more instructions on running the sample).

You can read all about the menu utility in the document [\psx\sample\module\menu\readme.txt](#). **This utility does not demonstrate the capabilities of the samples in \psx\sample\scea or \psx\sample\scee.**

**How it works:** This program works by installing itself in the upper 5 Megabytes of the Development Board's memory. The child sample programs, and their data are loaded within the first megabyte (or so) of memory, and then the function *Exec(...)* is called. Due to the memory placement, the main executable has no danger of being overwritten, and it can keep track of the menu status. Theoretically, it could be possible to rewrite the programs so that the Programmer Tools CD could be made into a commercial black PlayStation bootable CD. However, the PlayStation has only 2 Megabytes of RAM, and there are over 140 samples being produced, changed, and constantly updated, so in practice this is quite difficult to manage. In addition, the samples are accessed using the PC-File system. Now in theory, you should be able to run the samples directly from the CD-ROM, but due to the excessive size of the file hierarchy on the Programmer Tools CD-ROM (with over 3500 files and 377 folders), "CdSearchFile" doesn't work correctly. Work arounds are possible (refer to Ben Fawcett's article on CCS2POS on the Technical Reference CD), but frankly -- we're running out of time. So for now, we hope you like this solution.

To see an abbreviated version of this program for use in your own games, refer to [\psx\sample\module\execmenu\readme.txt](#)



**How do I run these samples?** The data for these movie samples is stored in the "\Xdata" directory on this CDROM, which needs to be placed in the DTL-H2010 or the DTL-H2510 prior to executing the samples. Alternatively, you can place the data onto the hard drive of your CD-Emulator. Follow the instructions for running CD samples [here](#).

**Warning:** With the release of libds in Library 4.0, libds.lib will be required for the operation of libcd.lib. Make sure that libds.lib is linked, even if you do not plan to use it. This measure has been taken care of for psyq.ini. As noted below, the new sample directory **\psx\sample\ds** illustrates how to use the new ds functions.

**Directory:** \psx\sample\cd\earth

This example streams movie data from the CDROM and places it on a spherical surface using RotPMD functions.

**Directory:** \psx\sample\cd\movie

- *tuto0* Simple streaming program. Time-out retry feature was added to the movie play sample program. This feature restarts playing the movie data picking up where it was left off in case it is halted due to, for example, opening a PS cover by causing *strNext()* to time-out.
- *tuto1*: Allows variable screen resolution.
- *tuto2*: Added memory streaming to *tuto0.c*
- *tuto3*: Same as *tuto0.c* but avoids frame skipping.

**Directory:** \psx\sample\cd\str3d.

Sample of the combination of moving pictures with streaming and 3D model display. Spreading loads of animation with DecDCTvlcSize()

**Directory:** \psx\sample\cd\tuto.

CD-ROM tutorials exercising the functions available in the library libcd.lib:

- *tuto0*: simplest CD-Player (polling type)
- *tuto1*: simplest CD-Player (interrupt type)
- *tuto2*: auto repeat play among 2 points of CD-DA
- *tuto3*: auto repeat play using CdldataEnd
- *tuto4*: fast operation using CdControlF
- *tuto5*: auto repeat play among 2 point of CD-XA.
- *tuto6*: interleaved audio/data channel
- *tuto7*: background CD read
- *tuto8*: multi file CdRead
- *tuto9*: load and execute programs
- *tuto10*: high level CD-ROM file access
- *tuto11*: test CD type

**Directory:** \psx\sample\ds\tuto.

A simple tutorial for using libds.lib, consisting of the following tutorials:

- *tuto0* - *tuto11*. These are the rewritten libcd tutorial for libds.lib. Since the structure of the program has been kept as in tact as possible, these are not necessarily the most appropriate methods for libds.lib, but by comparing these with the tutorial of libcd, you should be able to grasp the characteristics of libds.

- *tuto12*: The sample for the comparison of the time required until the playing of the music for the cases of using forward seek to play DA and not.
- *tuto13*: The sample for the comparison of the time required until the playing of the music for the cases of using forward seek to play XA and not.
- *tuto14*: The sample of comparison of the time required for continuous reading of multiple files (sequential recording on the CD and reading without seeking individual file at a time, but rather, continuously read in recorded sequence) and seeking and reading individual files at a time.

For more information, you can read this [readme](#) and similar topics on the Technical Reference CDROM. *June 1997*

**Directory:** \psx\sample\scea\cdemu

A basic tutorial on using the CD-Emulator and burning a gold disk, from start to finish. **Contributed by Chia-Ming Wang, Nov. 1997**

**Directory:** \psx\sample\scee\cd\cdplayer

Makes your PlayStation into a CD-Player. When you put a music CD into your DTL-H2010 or DTL-H2510, you can play the tracks. Contributed by members of SCEE **Dave Virapen, Allan Murphy, and others December 1995**

**Directory:** \psx\sample\scee\cd\movie2

A simple movie player module with the following features:

- Plays movies at 24 or 16 bit at any horizontal screen resolution.
- Pre-processing stage to find the maximum VLC buffer size.
- Well defined interface and self contained module that can be easily included into your program without too many problems.
- Efficient use of memory.
- Lots of error checking.

Contributed by **Vince Diesi**, June 1997.

**Directory:** \psx\sample\scea\cd\xaplayer

Shows how to use interleaved XA streams to store a large number of dialog lines and how to play them back nearly instantly. Unfortunately, we did not get the PACK1.XA and PACK2.XA files for Programmer Tools CD-ROM release 2.0. Contributed by **Buzz Burrowes**, Sony Interactive Studios America *October 1995*

**Directory:** \psx\sample\scee\demodisc

Source code hook for usage in the Demo cd used by the European developers. Please contact SCEE for more information on this. USA developers can SCEA, and we will forward your requests to SCEE. **Contributed by Allan J Murphy, Paul Holman, Vince Diesi, Richard Milner February 1997**

**Directory:** \psx\sample\module\cdexec

A module for executing PSX.EXE on a CD-ROM/CD-Emulator with DTL-H2000

---

## Scratch pad sample

**Directory:** \psx\sample\cmplr\scratch

Sample demonstrating use of the scratch pad area. Data is placed on the scratch pad area, and the difference of the processing speed can be seen with three different access methods.

**Warning:** Before you compile this program, you must edit the file "main.lnk" to match your environment in order to let the compiler know about "spad" group. Read the file [psx\sample\cmplr\scratch\readme.txt](#) for more information.

---



These samples demonstrate how to use the combat cable, lightgun, multitap, and others.

**Directory:** \psx\sample\control\mouse\

A mouse control sample that can process cursor-movement and mouse clicks.

**Directory:** \psx\sample\serial\comb\

Samples for the combat cable:

- *tuto1*. Sample for synchronous communication. **UPDATED**
- *tuto2*. Sample for synchronous write and asynchronous read.
- *tuto3*. Sample for asynchronous write and asynchronous read.

**Note** This sample program is for versions 3.7 libcomb and later, and will not run with any previous versions of libcomb. Sample programs for version 3.6 and earlier versions of libcomb will not run with Version 3.7 libcomb.

**Directory:** \psx\sample\control\gun

Sample program for gun controller. When gun controller is attached to the PlayStation and the trigger is pulled, the position where the gun controller is facing is displayed on the screen. **Note:** DTL-H2500 users may have had problems running the light gun samples in the past. Place the new sn.bin (075) in the directory \psn\bin, and while in the directory, run the program "pflash.bat". This will fix the problem.

**Directory:** \psx\sample\control\mouse

Sample to process cursor-movements and clicking with a mouse

**Directory:** \psx\sample\serial\sio

- *tuto1*. SIO driver sample to connect the debugging station with PC via H3050 and echo-back input from PC using standard I/O drivers, and callback is implemented by a receive interrupt.
- *tuto2*. SIO driver sample to connect the debugging station with PC via H3050 and echo-back input from PC **without** using standard I/O drivers, and callback is implemented by a receive interrupt.

**Directory:** \psx\sample\control\tap

Multi tap sample

**Directory:** \psx\sample\scea\cntrl

Controller demonstration demonstrating the controller API. Sprites are generated according to the controllers attached, such as a pad-sprite if a pad controller is being used, or a gun-sprite if a light-gun controller is in use. Written by **Mike Fulton** of SCEA.

**Directory:** \psx\sample\scee\etc\mtap

Multitap example from the European Developers conference, 1996. Shows you how to read data from the multitap. Please contact SCEE for more information on this. USA developers can contact SCEA, and we will forward your requests to SCEE

**Directory:** \psx\sample\scee\etc\control

This is a general peripheral tester that demonstrates the use of

- The Standard Controller.
- The Mouse.
- The NegCon.
- The Analog JoyStick.
- The New Analog Controller.
- The Multi Tap.

Note that the MultiTap only gives the buffer readout. To run the code correctly just type `runit.bat`. This will cause some pqblooms of the tims. Contributed by **Kevin Thompson** at SCEE *June 1997*

**Directory:** \psx\sample\scee\etc\gun

This is a peice of code to show the use of the Konami Gun and the Gcon.45. To run, type “`runit.bat`” to get the code running and then shoot the planet. Note: If the gun seems inaccurate, raise the brightness of the TV and avoid a bright light. Written by **Kevin Thompson** of SCEE.

---

## DMPSX Samples

Just what is DMPSX? Inline macros have been created in the hader files *inline\_a.h*, *inline\_c.h*, and *inline\_o.h* to take the place of the regular GTE functions. DMPSX will process the macro headers and produce GTE function calls in assembly language, to avoid the overhead of making a function call and to help your code fit within the 4KByte Instruction Cache of the R3000.

**Directory:** \psx\sample\graphics\dmpsx

"tuto0.c" shows an example of improving program performance using DMPSX You can compare the performance of two programs, one using the libgte low-level- functions and another using the DMPSX. This sample program runs the DMPSX version as a default. In order to modify the program to a libgte low-level-function version, just comment out the first define statement

```
#define DMPSX_MACRO
```

The number displayed on the screen represents the total time (in horizontal sync's ) of the calculations and drawings.

**Directory:** \psx\sample\scea\gte

An example of GTE optimization using DMPSX, contributed by **Mike Acton**, of Sony Interactive Studios America in San Diego (contributed October 1997). In Mike's own words: "Visually there is nothing stunning being displayed. The intent is to demonstrate a few optimization methods using the

GTE. " Thanks Mike for the superb documentation!

---

## Graphics Samples

The graphics samples demonstrate a broad range of techniques for programming with the GPU and the GTE. The directory `\psx\sample\graphics\data` contains data used by some of the example programs. To learn how to run the samples, click [here](#)

**Note:** Some of the samples may be confusing to analyze because there are no ".tim" files that are loaded -- there's merely a header with a large array of unsigned chars. These header files **are** tim files that have been converted from binary format. Its advantageous to load static data this way during development because the compiler keeps track of the addresses for you.

**Directory:** `\psx\sample\graphics\basic`

A menu based program that launches the other basic samples:

- 2D -- Exercises functions of `libgpu.lib` by texture mapping a 512x256 texture pattern on a 3-dimensional curved surface. When 'select' is pressed, the pattern will come to pieces.
- Balls -- Exercises functions of `libgpu.lib` by displaying a lot of 16x16 sprites. Demonstrates usage of the functions `FntPrint()`, `KanjiFntPrint()`, `VSyncCallback()`, and `VSync()`
- Diffuse -- Demonstration of GTE performance using `libgte.lib` functions. 16x16 balls or rectangle polygons are diffused from the origin in the world coordinate system.
- mat -- "Matchang" sprite animation. The Matchang animation is placed in the 3rd dimension
- rgb24 -- Demo in the 24-bit mode of the GPU of the PlayStation. Remember, you cannot use the GTE functions in this mode. This example uses the following functions: `StoreImage()`, `LoadImage()`, `MoveImage()` .

**Directory:** `\psx\sample\bg`

BG drawing function sample using functions in `libgs.lib`. TIM\CEL\BGD files made up with the Sprite Editor (available on the Graphic Artist Tools CDROM) may be read in and displayed.

- *bgsample*: Background sample.
- *fix32*: Background sample (fast).

**Directory:** `\psx\sample\graphics\clutfog`

Fog sample with clut.

- *tuto0*: A clut is generated frame by frame, and transferred to the vram.
- *tuto1*: Some cluts are placed on the vram, and switched according to the depth of fog.
- *tuto2*: Cluts are switched by DR\_MOVE. Applicable to drawing by `libgs` as well.
- *tuto3*: textured polygon with depth queue (by exchanging CLUT with DR\_LOAD primitive)"

**Directory:** `\psx\sample\graphics\divide`

- *clip*: Divide function examples to avoid texture distortion. In `readme.txt`, PCpoly function examples included.
- *active*: Sub-division sample with the direct mapping. Crack problem and z-sorting by the maximum value not by the center of gravity are included.

**Directory:** `\psx\sample\graphics\fballs`

Sample based on "balls" program for decreasing the drawing time. Uses the *libgpu.lib* functions.

**Directory:** \psx\sample\graphics\gsgpu

Sample using functions in both libgs.lib (the higher order 3D graphics package) and libgte.lib (the primitive 3D graphics package) together.

- *tuto0*: Uses AddPrim() in libgs.
- *tuto1*: Draws libgs objects with DrawTag().

**Directory:** \psx\sample\graphics\HMD\anim

Animation examples using the new HMD format.

- *tuto0*. General HMD animation
- *tuto1*. Realtime Motion Switch #1
- *tuto2*. Realtime Motion Switch #2
- *tuto3*. General HMD animation viewer using view point animation.
- *animview*. General HMD program (This sample is used to view a vast array of HMD data, located in the \psx\data\hmd directory).

**Directory:** \psx\sample\graphics\HMD\basic **UPDATED**

Basic viewers using the HMD format

- *tuto0*. General HMD viewer
- *tuto1*. HMD viewer using shared polygons
- *tuto2*. Shuttle viewer using HMD format
- *tuto3*. HMD viewer using sub divide

**Directory:** \psx\sample\graphics\HMD\Common **UPDATED**

Scanning routine for HMD equipment primitive.

**Directory:** \psx\sample\graphics\HMD\Mime

Demonstrating of using HMD with MIME.

**Directory:** \psx\sample\graphics\HMD\pdriver

Sample code of the primitive drivers for HMD.

For more information, click [here](#). June 1997

**Directory:** \psx\sample\graphics\jimen

Function sample for undistorted texture mapping, demonstrating the use of functions in libgte.lib.

**Directory:** \psx\sample\graphics\mesh\qmesh

Two dimensional mesh

- *tuto0*: sample of QMESH function...screen clip
- *tuto1*: sample of QMESH function...terrain data

**Directory:** \psx\sample\graphics\mesh\rmesh

Illustrates a round mesh

**Directory:** \psx\sample\graphics\mesh\smesh

Illustrates strip meshes:

- *tuto0*: Drawing performance of SMESH functions
- *tuto1*: Browsing SMESH functions' drawing mode
- *tuto2*: Icosahedron

**Directory:** \psx\sample\graphics\mime\joint

MIME Interactive Animation . MIME sample program with GsDOBJ5. Controlling 4 MIME parameters with L1,L2,R1,R2 buttons. The data is a simple gouraud-shaded polygon, and MIME processing is performed for the normal.

**Directory:** \psx\sample\graphics\mime\vertex

Sample code for MIME using GsDOBJ5. Controg 4 MIME parameters with L1,L2,R1,R2 buttons. The data is a simple Gouraud-shaded polygon, and MIME processing is performed for the normal.

**Directory:** \psx\sample\graphics\mime\vmime

This is a sample program for manipulating an articulated model using the vertex multiple inbetweening method and/or the joint multiple inbetweening. They demonstrate two kinds of joint-MIMes: *axes-MIME*, which is interpolation by axes , and *RPY-MIME*, which is interpolation by roll, pitch, and yaw angles.

- *tuto0*: shows an example of interpolation using both joint-MIME and vertex-MIME. tuto1 is also able to manipulate states of the model.
- *Tuto1*: shows an example of interpolation using only joint-MIME.

**Directory:** \psx\sample\graphics\mipmap

Sample demonstrating mipmaps.

**Directory:** \psx\sample\graphics\misc\60frames

Demontstrates the difference between 60 frames from 30 frames.

**Directory:** \psx\sample\graphics\misc\getode

Uses Vsync with Interlace Mode. This example addresses a problem that developers encountered when trying to achieve 60 fps in interlaced mode. Briefly, if you are using the interlace single buffer(vertical-480 dot-mode) AND the drawing switch is done by VsyncCallback() instead of Vsync(), only the odd number fields are cancelled. Thus this would cause the residual image problem since the background screen is not completely cleared.

In order to avoid the problem, drawing must be started immediately after V-BLNK completion. Unfortunately, since there is no mechanism available to detect the completion of the V-BLNK using the current VSyncCallback(), it is necessary to either

- - count the number of H-Sync using VSync()1 during V-BLNK  
or
- - add new callback using H-Sync callback (RCnt2).

We would like to introduce more certain way to solve the problem by disclosing the function called, GetODE() to check if the current field being displayed is the odd or even fields. GetODE() was officially included in the Release 3.7. This example program illustrates how to use the GetODE( ) function.

For more information, click [here](#).

**Directory:** \psx\sample\graphics\oden

Oden Moving 3 light sources interactively, changing their colors, performing the real-time lighting calculation.

**Directory:** \psx\sample\graphics\phong

An example showing how to do Phong shading using functions in *libgte*

**Directory:** \psx\sample\graphics\ppm

Undistorted mapping (perfect perspective mapping) using functions in *libgte*

**Directory:** \psx\sample\graphics\rotate

Examples demonstrating rotation.

- *arot*: Rotation angle interpolation program
- *intrpol*: various kinds of interpolating about rotation
- *mat2rot*: Get Euler's angles from rotation matrix

**Directory:** \psx\sample\graphics\rotmat

Difference between RotMatrix and RotMatrix\_gte

**Directory:** \psx\sample\graphics\shadow

This program calculates a shadow cast by a floating triangle. Clipping is performed accurately, so the shadow can be cast on objects such as stair steps.

**Directory:** \psx\sample\graphics\tmdpmd

TMD/PMD data viewer.

**Directory:** \psx\sample\graphics\tmdview\lowlevel

Lowlevel sample with *GsTMDfast...*() functions

- *tuto0*: 3 sided polygon, flat
- *tuto1*: 4 sided polygon, gouraud
- *tuto2*: Eliminate a gap between polygons (with dmplx)
- *tuto3*: Mipmap version (with dmplx)

**Directory:** \psx\sample\graphics\tmdview\rcube

Rotating cubes. Variable effect samples for 3D

**Directory:** \psx\sample\graphics\tmdview\shuttle

Hierarchical coordinate system sample with a space-shuttle model. The Animation such as opening/closing the hatch is displayed by setting the hatch and a satellite in the shuttle in its child coordinate.

**Directory:** \psx\sample\graphics\tmdview\tmdview3

The simplest PMD data display program with GsDOBJ3.

- *tuto0*: Simplest TMD data display program with GsDOBJ3.

**Directory:** \psx\sample\graphics\tmdview\tmdview4

The simplest TMD data display program with *GsDOBJ2*

- *tuto0*: simple tmdviewer using GsDOBJ2(GsSortObject4())
- *tuto1*: using GsSortObject4J()
- *tuto2*: active sub divide sample
- *tuto3*: sample code for split screen using GsDOBJ5
- *tuto4*: sample code for multi ot and using same object with different hadlers.
- *tuto5*: multi screen coordinate sample
- *tuto6*: sample code of subjective move.
- *tuto7*: using GsSortObject4J() and using material attenuation in GsDOBJ2

**Directory:** \psx\sample\graphics\tmdview\tmdview5

TMD data display program with *GsDOBJ5*

- *tuto0*: Simplest TMD data display program with GsDOBJ5
- *tuto1*: Sample of split screen
- *tuto2*: With modeling data some objects are displayed. More than one OT are used.
- *tuto3*: Automatic division with GsDOBJ5 attribute
- *tuto4*: Multi-screen coordinate system
- *tuto5*: Sample rewritten with GsSortObject5J
- *tuto6*: Sample where the viewpoint is moved subjectively

**Directory:** \psx\sample\graphics\tod

Animation with tod

- *janken*: Multiple interactive tod animation
- *todview*: Simple animation

**Directory:** \psx\sample\graphics\trr

Samples of how to use the *TransRot...()* functions to eliminate gaps between polygons.

**Directory:** \psx\sample\graphics\tuto

- *tuto0*. Displaying sprites
- *tuto1*. Drawing test with OT
- *tuto2*. Drawing a rotating polygon with GTE
- *tuto3*. Drawing a rotating cube
- *tuto4*. Drawing a cube with the light source
- *tuto5*. Drawing multiple 3D objects
- *tuto6*. Testing a 1D scrolling BG
- *tuto7*. Drawing a cube with the depth cueing
- *tuto8*. Showing a cell-type BG
- *tuto9*. Showing a 3D-cell-type BG
- *tuto10*. 3D cell type BG (bird view)
- *tuto11*. pseudo mosaic effect
- *tuto12*. pseudo line scroll effect
- *tuto13*. multi window operation

**Directory:** \psx\sample\graphics\walk

An object walks on a polygon Constraining an object on a polygon. On a object (object1) created by polygons, another object (object2) may move around. The object1 may take any shapes. The object2

changes its direction according to the direction of the object1's normal.

**Directory:** \psx\sample\graphics\texaddr\wave

This sample shows the pseudo environment map of the wave reflection of a clock by modulating texture address in real time, and the refraction of a clock under the water. tuto1, tuto2 and tuto3 are tuned with DTL-H2700 and Performance analyzer. "main" has auto-demo mode that changes three scenes.

- tuto1. clock in a swimming pool with DMPSX
- tuto2. clock in a swimming pool tune CPU process (scratch pad is assigned to stack)
- tuto3. clock in a swimming pool tune GPU process (divide OT)
- main . 2-minute auto demo including three scenes

**Directory:** \psx\sample\graphics\zimen

Terrain. A group of programs to display the endless plane

- *tuto0*: Active primitive subdivision (with dmpsx)
- *tuto1*: Basic viewing volume clipping
- *tuto2*: Meshed ground pattern without height
- *tuto3*: Meshed infinite ground pattern
- *tuto4*: Meshed ground with active subdivision
- *tuto5*: Terrain sample with CLUT FOG (version with libgs)

**Directory:** \psx\sample\scea\fire

This is an instructional program that illustrates how to make a procedural texture which can then be used multiple times. Contributed by **Mike Koziniak** of SCEA, November 1997. A modification of the \sample\scea\graphics\flame demo by Jason Page, SCEE.

**Directory:** \psx\sample\scea\graphics\chrome2

A stunning example of environment mapping. A spinning glass crystal is surrounded by some chrome objects. The objects look so real, you'll be stunned. This demonstrates the best in PlayStation capabilities. Contributed by **Dave Virapen** at SCEE, *June 1997*.

**Directory:** sample\scea\graphics\flame

Stunning example of generating real-time flames on the PlayStation. Contributed by **Jason Page** at SCEE, April 1997.

**Directory:** sample\scea\graphics\fractal

Eye-popping Fractal landscape viewer. Contributed by **Jason Page** at SCEE, December 1997.



**Directory:** \psx\sample\memcard\card

This sample program uses a menu to perform various kinds of accesses on the slot 1 (left) memory card, such as state-monitoring, formatting, and creating.

**Directory:** \psx\sample\memcard\mcrd

Memory card samples illustrating the use of libmcrd.lib.

- *tuto0*.Synchronous processing: Icon generation program
- *tuto1*.Asynchronous processing: Sample 1 Displays the list of files on the memory card
- *tuto2*.Asynchronous processing: Sample 2 Selectively copies the Slot1 file(s) to Slot 2

**Directory:** \psx\sample\old\etc\cman

Memory card management sample which exercises memory card file utilities such as state-monitoring, formatting, creating.

**Directory:** \psx\sample\memcard\dongle

Illustrates how to use the memory card as a dongle.

**Directory:** \psx\sample\scee\etc\archive

A piece of code to perform a memory card save to a PC Hard drive and vice versa. 20 saves have been supplied with the code for you to test your memory card code with, including a large variety of European and Japanese Saves to mix ASCII and Kanji fonts in the titles. *Contributed by Kevin Thompson, SCEE, November 1997*

**Directory:** \psx\sample\scee\etc\card

Memory card access. Lists the contents of the memory card inserted in the controller box DTL-H2080. **Note:** *This uses routines from libraries earlier than Library 4.0, which may require different routines.* Contributed by **Dave Coombes** , SCEE February 1996>

**Directory:** \psx\sample\scee\etc\cardconf

Memory card access from SCEE developer's conference. You will have to change the ".lnk" file to suit your development environment. It allows you to read and write from memory cards using optimized techniques, while displaying animated graphics and playing seq data. **Note:** *This uses routines from libraries earlier than Library 4.0, which may require different routines.* Contributed by **Dave** at SCEE, February 1996

**Directory:** \psx\sample\scee\etc\cman41

A piece of code to demonstrate how to use Memory card's/Multitaps and Midi music. Compatible with Lib's 3.7 and above. Contributed by **Kevin Thompson** with code from **Allan J. Murphy** at SCEE June 1997

**Directory:** \psx\sample\scee\etc\memcard

Kev writes: *"This library is something I coded to make programming the memory cards as easy as possible using the low-level functions supplied libcard.lib. I have also supplied the source code to the library. this is so you can see just how the code is working and if needed alter and re-compile."* Contributed by **Kevin Thompson** at SCEE, November 1997.

**Directory:** \psx\sample\old\etc\card

Memory card boot sample

---



**Directory:** \psx\sample\kanji\asc2sjis  
Converts the ASCII code to the Shift-JIS code

**Directory:** \psx\sample\kanji\fontdata  
Kanji font data. Size and types available are:]

- 11/13/15 dots.
- non-kanji/first level/second level/vertical/half-size characters.
- Code conversion table

**Directory:** \psx\sample\kanji\kanjdiv  
Command to extract character data. Extracting image data from font data in character units.  
Command and viewer for extracted image data.

**Directory:** \psx\sample\kanji\kanjifnt  
Sample to use font data by size.

**Directory:** \psx\sample\kanji\sjiscode  
KANJI Code Viewer Program. Shift-JIS codes of the built-in fonts can be displayed.

---

## Modules, menus, and overlay samples

If you need to have a menu launcher, or you want to partition your game in modules, use the examples here. These show you how to run overlays, or execute programs using "LoadExec( )".

**Directory:** \psx\sample\module\execmenu  
Two examples showing how to launch a program using calls similar to "system( )". The programs launch a menu which allows you to selectively launch the executables in the subdirectories of BALLS, RCUBE, and ANIM. (These are modified, non-standalone versions of the BALLS, RCUBE, and ANIM appearing in the directories \psx\sample\graphics and \psx\sample\movie. You will have to create a CD-ROM image on either a gold disk or the cd-emulator according to the directory layout in the [readme.txt](#) file.) The sample in this directory achieves the results through the use of the functions LoadExec() and Exec(). An alternative method, using overlays, appears in [psx\sample\module\overmenu](#).

**Directory:** \pssn\sample\overlay  
This example consists of a main program and 3 overlaid levels. The example is only a shell to demonstrate how to achieve overlays. This is a fine example to start learning how to do overlays; after mastering this sample, you can then try out the \psx\sample\module\overmenu sample.

**Directory:** \psx\sample\module\overmenu

Using the technique of overlays, this programs launches a menu which allows you to selectively launch the executables in the subdirectories of BALLS, RCUBE, and ANIM. (These are modified, non-standalone versions of the BALLS, RCUBE, and ANIM appearing in the directories \psx\sample\graphics and \psx\sample\movie. You will have to create a CD-ROM image on either a gold disk or the cd-emulator according to the directory layout in the [readme.txt](#) file.) Alternative methods, using *LoadExec()* and *Exec()* can be found in [psx\sample\module\execmenu](#)

**Directory:** \pssn\sample\prefsmpl

An example showing how to use the tool "prefsect", which renames the sections in a psy-q object file by prefixing the names with a defined string. For more infomation, click [here](#).

**Directory:** \pssn\sample\overlay

An example from SN Systems for showing how to do overlays using psylink. For more infomation, click [here](#).

**Directory:** \psx\sample\module\cdexec

Start-up utility from CD-ROM/Emulator, used when activating from CD-ROM/Emulator on DTL-H2000 with snpatch executed. Alternative module of 'resetps 0'.

**Directory:** \psx\sample\module\menu

Sample Program Viewer which loads execution file. Sample execution files are activated from the menu. It is necessary that the program which can be activated from this menu should link "none2.obj" and be written in "menu.lst".

**Directory:** \psx\sample\graphics\screen

Screen Frame buffer viewer. Demo to explain the display mode and display environment parameters.

---

## MPEG compression/decompression

**Directory:** \psx\sample\press\tuto

Tutorial using the functions in the MPEG decompression utilities in the library *libpress.lib*.

- *tuto1*: simple VLC decode and MDEC on memory decompression
  - *tuto2*: parallel execution of *LoadImage()* and *DecDCTout()*
  - *tuto3*: simple on-memory movie operation
  - *tuto4*: handshake using callback
  - *tuto5*: parallel execution of *LoadImage()* and *DecDCTout()* using callback.
  - *tuto6*: complete background on-memory movie decompression
  - *tuto7*: fine tune-up for parameters
- 

## SCEA Samples

These samples were created by members of [Sony Computer Entertainment America](#). In addition, other contributed samples have been incorporated into the categories above.

**Directory:** \psx\sample\scea\pcfs

PC file -system demonstration Demonstrates how to use the PC-file system function calls using the

Psy-Q development environment. This enables you to write to your PC's hard-drive. Written by *Mike Fulton* of SCEA.

---

## SCEE Samples

The samples in this directory were contributed by members of [Sony Computer Entertainment Europe](#) and European developers. In addition, other contributed samples have been incorporated into the categories above.

**Directory:** \psx\sample\scee\etc\PAL

Example for PAL users. Please contact SCEE for more information on this. USA developers can contact SCEA, and we will forward your requests to SCEE

**Directory:** \psx\sample\scee\kcheats

Complete source for the "killer cheats" program. You must burn a CD or load the CD emulator with the appropriate files. Please contact SCEE for more information on this. USA developers can contact SCEA, and we will forward your requests to SCEE.

**Directory:** \psx\sample\scee\subdiv

Alternative polygon subdivision routines. This does a fast texture mapping to flat and shaded triangles and quadrilaterals. Contributed by *Derek Leigh-Gilchrist*.

**Directory:** \psx\sample\scee\utils\except

Example exception handler that is compatible for versions 3.1 and 3.2 of the library. It is pretty old. USA developers can contact SCEA, and we will forward your requests to SCEE. Contributed by *Brian Marshall*

**Directory:** \psx\sample\scee\utils\profil

Example code profiler for timing analysis. You will have to edit the "protest.lnk" file to match your development environment.

---

## Serial Cable samples

On the blue debugging PlayStation, a serial port can be used to communicate with the serial port of an IBM PC-AT compatible, via the DTL-S3050 cable. For these samples to work, you should burn a gold disk. Read the instructions in [\psx\sample\serial\sio\tuto1\readme.txt](#). Full source for a downloading program (which can download your program and then execute it) can be found in [\psx\utility\pcdown](#); it uses ideas from these samples.

**Directory:** \psx\sample\serial\sio\tuto1\tuto1

Sample to connect the debugging station with PC via H3050 and echo-back input from PC.

**Directory:** \psx\sample\serial\sio\tuto1\tuto1

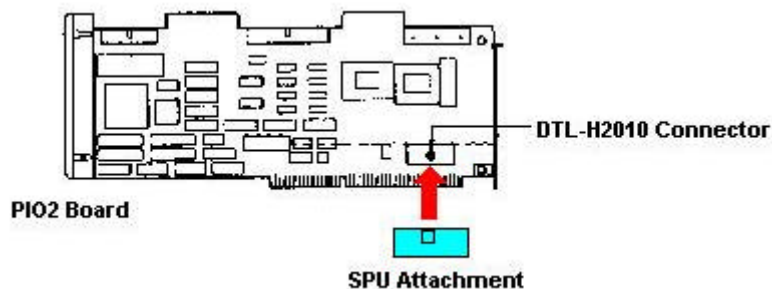
Sample to connect the debugging station with PC via H3050 and echo-back input from PC

---



**NOTE:** Some sound examples available in `\psx\sample\sound` do not have any graphics. You will be faced with a black screen, but don't panic. Your controller pad serves as a musical keyboard, so just start pressing the buttons. Full instructions on the pad assignment are available in the "readme.txt" files in each directory.

**Warning:** If you have **trouble** hearing any sounds on a DTL-H2000, but you can see the graphics, one possible reason is as follows: If the DTL-H2000 is not connected to the DTL-H2010 (CD-ROM drive, sold separately), or the Psy-Q PSX04 (CD Emulator), the SPU may not reset. In order to insure proper functioning, attach the included blue SPU attachment to the DTL-H2010 connector:



Now for some samples!

**Directory:** `\psx\sample\sound\3DEffect`

This sample shows off 3d sound key on series. Best performance of 3d sound location can be heard with headphones.

**Directory:** `\psx\sample\sound\balls`

Example of combining sound and graphics. While pressing a button, balls bounce in the screen. When they hit against the wall, different sounds for each ball are generated. SEQ data is used as the background music. **Directory:** `\psx\sample\sound\basic`

Basic play 1. SEQ/SEP data processing function sample. SEQ and SEP data may be played simultaneously. SEP data consists of 3-connected SEQ data.

**Directory:** `\psx\sample\sound\cdvol`

SPU decoded data reading sample using routines in *libspu.lib*. Music played on the CD is read as the "SPU decoded data" from the SPU in the background, and the volume is displayed with a graph (with the display of the peak level).

**Directory:** \psx\sample\sound\mutual

Wave form data divided transfer sample using function in *libsnd*. At a timing divided wave form data is read into the main memory, and transferred to the sound buffer. This process is repeated until all parts are transferred. As a result, 2 pieces of music may be played.

**Directory:** \psx\sample\sound\simple

This is a sample program using the sound library. It plays the SEQ data file and displays tempo, volume, and status on the screen using the jump-table.

**Directory:** \psx\sample\sound\tuto

- *tuto1*: Pitch designation/key-on/key-off. According to the control-pad operations, a sound is played with variable pitches.
- *tuto2*: Mute. Performing the sound generation, mute-on, mute-off
- *tuto3*: SPU interrupt. Setting a interrupt in the middle of piano sound data. When the piano sound starts, and the interrupt occurs, a sine wave is generated.
- *tuto4*: Noise sound source. Generating a sine wave and noise by changing a pitch.
- *tuto5*: Divided transfer of wave form data. Alternating divided transfer and sound generation after the transfer in 2 voices.
- *tuto6*: Reverb. Generating a piano sound and designating 9 kinds of reverbs for the sound.

**Directory:** \psx\sample\sound\stream

SPU streaming sample program Sample using the SPU streaming library included in the basic sound library. Performs playback of 7-channel (14 voices) SPU streams by operating the control pad. Displays the state of SPU streaming on the screen.

- *tuto1*: The background is "balls".
- *tuto2*: The background is "movie".

**Directory:** \psx\sample\sound\xse

Auto-effect. Example using the sound utility functions. Such effects as pitch-bend, auto-panning, auto-volume are produced to the keyed-on sounds. By moving a thumb in the scroll bar, auto-panning, auto-volume, and pitch-bend may be available.

---

## Thread Samples

**Directory:** \psx\sample\thread

Sample to process other jobs until the next VSync

---

## Math Samples

**Directory:** \psx\sample\math\tree

Sample demonstrating how to use the trigonometric math functions to draw a tree curve.

---



**We appreciate your comments and suggestions about our HTML documentation project.** Contact us at [DevTech\\_Support@playstation.sony.com](mailto:DevTech_Support@playstation.sony.com)

Copyright © 1998 Sony Computer Entertainment America Inc. All Rights Reserved.

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