Hard Disk Drive Usage Rules (For SCEI titles only)

Table of Contents

About This Manual

This is the Runtime Library Release 2.4 version of the *Hard Disk Drive Usage Rules* manual.

It describes the usage rules for the hard disk drive and applies to SCEI titles only.

Changes Since Last Release

None

Related Documentation

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

Typographic Conventions

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

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

Developer Support

Sony Computer Entertainment America (SCEA)

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

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

Sony Computer Entertainment Europe (SCEE)

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

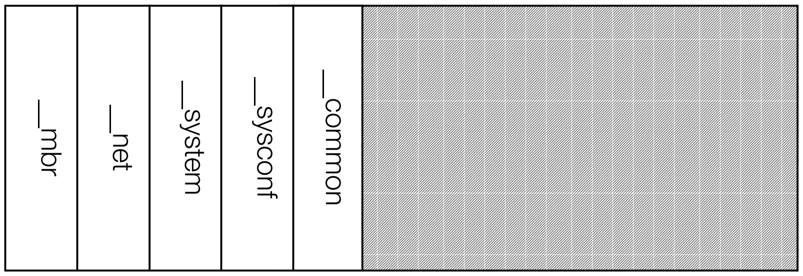
Order Information	Developer Support
In Europe:	*In Europe:*
Attn: Production Coordinator Sony Computer Entertainment Europe 30 Golden Square London W1F 9LD, U.K. Tel: +44 (0) 20 7859-5000	E-mail: ps2_support@scee.net Web: https://www.ps2-pro.com/ Developer Support Hotline: +44 (0) 20 7859-5777 (Call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

Hard Disk Drive Partitioning

Immediately after hard disk drive formatting, the drive must be partitioned as follows.

All system-partition IDs are preceded by two underscore characters ("_").

Figure 1



__mbr, __net, __system Partitions

Used by the system. Access by applications is prohibited.

__sysconf Partition

The system-configuration partition. Contains shared configuration files for mouse, keyboard, networking, Japanese-language input, etc. Also contains Japanese-language input dictionaries. For more details, see System Configuration Partition (__sysconf).

__common Partition

The common partition. Used for data shared by applications. For more details, see Common Partition (__common).

Installing Applications

This section explains how to install applications on the hard disk drive. Applications install themselves, at which time, the resources the system will use must be created. When installing an application, the first step is to create a partition, after which the necessary files are created in the partition.

Creating a Partition

At least one partition must be created each time you install an application, and in some cases, there will need to be multiple partitions. These partitions are referred to as "application partitions."

In the description below, the whole partition is indicated when the main partition is extended by sub-partitions.

ID Nomenclature

Partition IDs follow this format:

Up to 32 characters, using the period (".") as a delimiter.

```
PP.SLPS-00000.MAGIC.APPLICATION
(1)    (2)     (3)     (4)
```

1. PP (prefix)

   File system: P=pfs

   Partition parent/child relationship: P=parent; C=child

   Currently only PP and PC are supported.
2. SLPS-00000 (product code)
3. MAGIC (magic number)

   Zero or more characters from: A-Z 0-9 _
4. User-defined

   Zero or more characters from: A-Z 0-9 _

ID Usage

Based on the assigned ID, the system can make the following assumptions

- For a single application, mutiple partitions can be created with the same (2) field.
- Partitions with a PP prefix are handled as parents; those with PC are handled as children.
- A PP partition can have zero or more child partitions. These child partitions will have fields (2) and (3) identical to the parent's. Multiple PC partitions are differentiated by their (4) fields.
- A single parent/child relationship is displayed with a single icon. As such, resources will be needed to display the icon for the PP partition (for details, see "Resource configuration").
- When multiple parent/child relationships exist, they are differentiated with the (3) field, in which case, there will be an equal number of icons displayed.
- When the system deletes a partition, it will delete all the partitions with a parent/child relationship at the same time (note that this functionality is not in the library).
- A PC partition that does not have a PP partition is treated as corrupted.
- Parent/child relationships lacking resources for icon display are treated as corrupted.
- Note that the function of parent/child relationships of partitions described here is different from that of main/sub-partitions of extended partitions.

Example IDs

PP.SLPS-00000..APP_SYSTEM	...(1)
PC.SLPS-00000..APP_DATA0	...(1)
PC.SLPS-00000..APP_DATA1	...(1)
PP.SLPS-12345.SYSTEM.PAT0	...(2)
PC.SLPS-12345.SYSTEM.PAT1	...(2)
PP.SLPS-12345.STAGE1.RECORD	...(3)
PC.SLPS-12345.STAGE1.CAR_DATA	...(3)
PP.SLPS-67890..APP_NAME	...(4)
PP.SLPS-77777	...(5)
PC.SLPS-11111.FOO.HEHE	...(6)

The system will display six icons (1)(2)(3)(4)(5)(6). Because there is no parent partition for (6), it cannot display that icon, instead, the system will display a "broken" icon. Same-numbered partitions will be deleted at the same time when deleting system partitions.

Partition Size

Partition sizes are restricted as follows:

- A partition can be extended by no more than 1 GB. Note that this number is subject to future change.
- When installing an application, requests to create or extend a partition must be made in order of decreasing size. Failing to follow this rule will consume more drive space than needed, and may cause the application to be incorrectly installed.

Passwords

The use of passwords is optional but recommended. If a password string is entered into a program asis, it would be easy to obtain these passwords through a program dump. Some form of internal password scrambling is needed to prevent this.

Resource Configuration

This section explains how to configure resources needed for displaying icons or launching applications that are installed in the system.

Partition Attribute Area

Slightly under 4 MB of a partition is dedicated for the attribute area. Resource files are written to this area. The partition attribute area is only needed for PP partitions. Do not write anything to this area in a PC partition.

Because this area has no filesystem, it requires a header for management, which is formatted as:

header x 1 + resource file x 3 or 4

"Resource file" is the catch-all term for files such as system.cnf, icon.sys, list-view icons (icon files for deletion).

The system.cnf File

This is the same as the system.cnf file on a CD or DVD. It is a file containing statements relating to application start-up. The system.cnf file on the hard disk should contain the same information as the system.cnf file on the CD or DVD. A new item has been added, however: HDDUNITPOWER.

HDDUNITPOWER= NICHDD/NIC

- Item: HDDUNITPOWER

  When the system starts up, the network adapter/hard disk drive will be powered on. This item can be used to indicate whether the network adapter/hard disk drive should be powered off when an application is launched, or if the power should remain on. In the latter case, the hard disk drive's setup time will be shortened after the application starts. If this item is omitted, the network adapter/hard disk drive will be powered off.

- Setting value: NICHDD

  The network adapter/hard disk drive will not be powered off when the application is launched. This is normally chosen when performing an installation on the hard disk drive.

- Setting value: NIC

  Place the hard disk drive in an idle state when the application is launched, but do not power off the network adapter.

Sample system.cnf File

```
--- the beginning of file ---
BOOT2 = cdrom0:\SLPS_123.45;1
VER = 1.00
VMODE = NTSC
HDDUNITPOWER = NICHDD
--- the end of file ---
```

The icon.sys File

Fulfilling the same role as the icon.sys files used on the PS2 memory card, this file is used for the display icon. Unlike the icon.sys file on a PS2 memory card, which has a binary format, this is a text file. The parameter settings are very similar; the differences are noted below:

- The code for a return can be LF or CR+LF.
- Characters are encoded using UTF8. For glyph details, see "Allowable Range of UTF8 Character Codes (Glyphs)."
- Character codes in the range 0x00-0x20, except for return codes, are ignored. However, 0x20 can be used in exceptional cases.
- Items begin with a newline, are followed by an "=", then the setting value. Spaces or tabs can be placed before or after the "=".
- Text strings begin with character codes outside the range 0x00-0x20, but can include spaces, and end with a return code. Do not insert leading spaces (they are ignored).
- For information on character font formats (fixed-width or variable width), see "Allowable Range of UTF8 Character Codes (Glyphs)."
- Except for the MAGIC NUMBER, items can be in any order.

Table 1

Item	Values	Notes
PS2X	none	MAGIC NUMBER. Must always appear at start of file.
title0	string	Title line 1, 1-16 chars. Must be at least 1 character.
title1	string	Title line 2, 0-16 chars.
Bgcola	0-128	Background's alpha value
bgcol0	0-128, 0-128, 0-128	Background's upper-left color (R,G,B)
bgcol1	0-128, 0-128, 0-128	Background's upper-right color (R,G,B)
bgcol2	0-128, 0-128, 0-128	Background's lower-left color (R,G,B)
bgcol3	0-128, 0-128, 0-128	Background's lower-right color (R,G,B)
lightdir0	-1.0000-1.0000, -1.0000-1.0000, -1.0000-1.0000	Direction of light source 0 (X,Y,Z)
lightdir1	-1.0000-1.0000, -1.0000-1.0000, -1.0000-1.0000	Direction of light source 1 (X,Y,Z)
lightdir2	-1.0000-1.0000, -1.0000-1.0000, -1.0000-1.0000	Direction of light source 2 (X,Y,Z)
Lightcolamb	0-128, 0-128, 0-128	Ambient color (R,G,B)
lightcol0	0-128, 0-128, 0-128	Color of light source 0 (R,G,B)
lightcol1	0-128, 0-128, 0-128	Color of light source 1 (R,G,B)
lightcol2	0-128, 0-128, 0-128	Color of light source 2 (R,G,B)
Uninstallmes0	string	Uninstall warning message, line 1. 0-60 bytes (UTF8).
Uninstallmes1	string	Uninstall warning message, line 2. 0-60 bytes (UTF8). Ignored if uninstallmes0 is 0 chars.
Uninstallmes2	string	Uninstall warning message, line 3. 0-60 bytes (UTF8). Ignored if uninstallmes0,1 is 0 chars.

- Title 0/1 displayed as title, similar to that of the memory card saved-data icon. Number of characters is limited. Note that the number of bytes is not limited. To avoid user confusion, we recommend using a section that is different from the saved data.
- See "Uninstall Warning Message" for details on uninstallmes0/1/2.
- There is no item for specifying the icon file.

Sample icon.sys File

```
--- the beginning of file ---
PS2X
title0 = Brave Man of the Dusk (UTF8)
title1 = ~Unexpected Adventure Book~ (UTF8)
bgcola = 64
bgcol0 = 20, 70, 90
bgcol1 = 20, 20, 60
bgcol2 = 20, 20, 60
bgcol3 = 20, 70, 90
lightdir0 =  0.5,  0.5,  0.5
lightdir1 =  0.0, -0.4, -1.0
lightdir2 = -0.5, -0.5,  0.5
lightcolamb = 31, 31, 31
lightcol0   = 62, 62, 55
lightcol1   = 33, 42, 64
lightcol2   = 18, 18, 49
uninstallmes0 = Will also delete your character's journey. (UTF8)
uninstallmes1 = (Saved data will not be deleted) (UTF8)
uninstallmes2 =
--- the end of file ---
```

List-view Icon, Deletion Icon File

This is the icon's model/animation data file. It is exactly the same as the icon data file found on the PS2 memory card. The list-view icon is used when viewing a list of applications; the deletion icon is used when an application is being deleted. One icon can serve both purposes.

Icons have the same format as those on PS2 memory cards, but if the same icon is used for saved-data on the memory card, users are liable to be confused. We recommend creating separate icons for the application and the saved data.

Header

Data used for managing and organizing the above files. Indicates how files are stored. Size is fixed at 512 bytes. Represented as a C-language structure:

```
typedef struct {
     char magic[9];
     char reserved0[3];
     int version;
     struct {
            int offset;
            int size;
     } file[4];
     char reserved1[464];
} _HEADER;
```

The function of each of the members is shown below.

Table 2

Member	Function
magic	Must be the nine characters "PS2ICON3D".
reserved0	Reserved. Must be filled with zeroes.
version	Must be zero.
file[4]	Indicates system.cnf, icon.sys, list-view icon, and deletion icon, in that order.
offset	Byte offset from the beginning of the file's partition attribute area. The attribute area is aligned on a 512-byte boundary, so this offset must be a multiple of 512.
size	File size, in bytes.
reserved1	Reserved. Must be filled with zeroes.

Sample Header

Assuming

system.cnf: 74 bytes

icon.sys: 462 bytes

list-view icon: 250K bytes

deletion icon: 200K bytes

the header will look like:

```
HEADER header;

strncpy(header.magic, "PS2ICON3D", 9); // magic
memset(header.reserved0, 0, 3);        // reserved0
header.version = 0;                    // version

header.file[0].offset = 512;           // offset value of system.cnf
header.file[0].size =   74;            // 74 bytes

header.file[1].offset = 512+512;       // offset value of icon.sys
header.file[1].size =   462;           // 462 bytes

header.file[2].offset = 512+512+512;   // offset value of list-view icon
header.file[2].size =   250*1024;      // 250K bytes

header.file[3].offset = 300*1024;      // offset value of deletion icon
header.file[3].size =   200*1024;      // 200K bytes

memset(header.reserved1, 0, 464);      // reserved1
```

Although the attribute area is slightly under 4 MB, make sure that all the above files are stored within the first 1 MB. Also, if you are using the same icon for the deletion as for the list-view icon, make sure the offset and size are the same as for the list-view icon. In this case, you do not need to create a separate deletion icon.

Writing Resources

The following describes the actual process for writing resources to the partition attribute area. From the start of the attribute area, contents must be ordered as follows:

0. header
1. system.cnf
2. icon.sys
3. list-view icon
4. deletion icon.

The attribute area can only be read and written in 512-byte blocks. Also, make sure that when writing, the buffer is aligned on a 64-byte boundary. Sample code follows. Also see the sample header.

```
// Buffer (64 bytes aligned)
char buf[BUF_SIZE] __attribute__ ((alignd(64)));

// Open partition
if((fd = sceOpen("hdd0:PP.SLPS-12345..APP,passwd", SCE_WRONLY)) <
0){
      return fd; // error
}

// Write header
if((ret = sceWrite(fd, &header, sizeof(_HEADER))) < 0){
      return ret; // error
}

for(i = 0; i < 4; i ++){
      offset = header.file[i].offset;
      size = (header.file[i].size+511)/512*512; // 512-byte blocks

      // Clear buffer to avoid writing garbage.
      memset(buf, 0, size);

      // Read file into buf
      // Seek (if not perfectly packed)
      if((ret = sceLseek(fd, offset, SCE_SEEK_SET)) < 0){
            return ret; // error
      }

      // Write
      if((ret = sceWrite(fd, buf, size)) < 0){
            return ret; // error
      }
}

// Close
if((ret = sceClose(fd)) < 0){
            return ret; // error
}
```

Writing Files

After writing resources, write the files needed for installation. See the "Application Partition" section for caveats and details.

Uninstalling Applications

Application installation is handled by the application itself, but uninstalling is handled by the system. When uninstalling an application, the system simply deletes that application's partition. To facilitate uninstalling by the system, applications must be designed to that they can be completely uninstalled by just deleting their partitions. Applications that cannot be completely uninstalled by deleting their partitions cannot rely on the system's uninstall function, and will need their own special uninstall function.

System-based Uninstall

When a user performs an uninstall by selecting an application icon, all partitions having parent/child relationships are deleted. The system handles uninstall via these partition deletions.

Uninstall Warning Message

When the system is deleting an application, if uninstallmes0/1/2 in the application attribute area of icon.sys is not NULL, it will display the warning message. The warning message can be used in the following situations:

- When saved data is in the application partition, and it will be deleted as part of the uninstall
- When you want to alert the user that there is other user data apart from saved data in the application partition, which will be deleted as part of the uninstall
- When there are files used by the application located in other partitions, which will remain after the uninstall
- When you are not using the system-based uninstall but wish to prompt the user during an application-based uninstall
- Any other situation where you want to display some sort of message.

The warning message will be shown immediately before deleting the partition, but up until the warning is displayed, the user must not be locked out. Once the user has confirmed his intentions, deletion can proceed.

You must design your applications so that the system will operate normally after the application partition has been deleted. Consequently, we recommend designing your applications so that they can be uninstalled simply by deleting their partitions.

The length of the warning message is limited to a certain number of bytes, not characters: each line must be 60 or fewer bytes, but can contain any text within that limit. Allow for some empty space on the left and right of the screen when the system displays the message because long lines will be broken in mid-string. Take this into account and adjust your string lengths accordingly.

Application-based Uninstall

When using the application to uninstall itself, there are no special limitations (prohibitions on uninstalling other applications notwithstanding). Whether a warning message is displayed or not is up to the programmer. There is no requirement that an application-registered message like uninstallmes0/1/2 be displayed. However, to avoid user confusion, we do recommend showing a warning message.

The system-based function that deletes all partitions with parent/child relationships at the same time is not available through a library. It is possible to determine which partitions are used by an application through the partition IDs, and you will need to use that information to delete them all individually.

Application Partitions

Application partitions have no particular restrictions on their file structures or directory structures, so applications can use them freely. However, the system has no function for browsing the contents of an application partition, which effectively prevents third parties from browsing the contents at will (we recommend using partition passwords for greater security in this respect).

Saved Data

When saved data is stored in the application partition, it can be written in any format, except that it must be done so that it can be deleted as part of the uninstall. We recommend using a format so that the data can also be saved to the PS2 memory card or the common partition (see Common Partitions (__common) for more information).

Program Files

For security reasons, you are currently prohibited from storing all or part of an executable program file on the partition. This includes elf files or overlays. In the future, we plan to publish a format that can be used for saving program files.

Access by Third Parties

Access to application partitions apart from the ones created by the application itself is prohibited in practice. It's as if no password had been set, so access is not permitted.

This restriction doesn't apply in the following situations:

- When you want to exchange data between different applications from the same vendor, or different titles in the same series.
- When two vendors have explicitly given permission for shared access.

Partition Expansion

If a partition turns out to be too small when the application is being executed, the partition can be expanded without restriction. This can be done either by a sub-partition extension or creating a new partition. When creating a new partition, you must follow the rules that apply to new IDs and resource configurations for new installations. If insufficient capacity remains for the expansion, the user should be notified. Also, in this case, we recommend a recovery procedure that will save data, etc.

Partition Contraction

Operations such as deleting a data file can leave behind significant empty space in the partition. In this case, the partition can be shrunk. This is a time-consuming process, though, so you should confirm with the user before proceeding.

Launching Applications From the System

The system has the ability to launch applications from the resource files of installed applications. This is done when the user selects a displayed application icon. The CD/DVD for the selected application needs to be in the drive for this to work.

When the user selects an icon, the system compares the filename of each BOOT2 item in system.cnf on the disk with that of the system.cnf partition resource file. If they agree it will launch the application from the disk. In this case,

```
cdrom0:\SLPS_123.45;1
```

will be written to argv[0], as usual, and

```
hdd0:PP.SLPS-12345..APP_MAIN:cdrom0:\SLPS_123.45;1
```

(Active hard disk drive device : partition ID : copy of BOOT2 item)

will be written to argv[1].

This is used to determine which partition's icon the user selected. When launching applications directly from the disk,

```
cdrom0:\SLPS_123.45;1
```

will be written to argv[0], as usual, and argv[1] is NULL.

Error Handling

If the filesystem encounters problems when reading or writing files, due to sharp shocks to the hard disk drive or the like, an access error will be generated. If the application cannot continue running, the user will be notified and the application terminated.

The next time the system is booted, a repair will be performed, and the partition attributes will be set to reflect that status. Because it is possible that files will be lost during a repair, we recommend that when an application mounts its partition, that it check the status and if it performs a repair operation, it should make sure that all installed files are uncorrupted. If it is difficult to check files for integrity, we recommend either prompting the user to do a rewrite/reinstall, or deleting the partition and then reinstalling. In either case, we recommend taking steps to prevent user data from being lost.

System Configuration Partition (__sysconf)

This partition is used to store common configuration files. Applications can access these files, and write to them as needed.

The following files and directories are created immediately after the system install.

Directories

atok/	Contains files used for Japanese text conversion.
font/	Contains font files.
icon/	Contains generic data icons.
etc/	Reserved for future use.

Details on these directories will be added in a future document revision.

conf/	Contains the two files described below, which are created immediately after the system install.

conf/system.ini

Contains settings for the mouse, keyboard, Japanese input, etc.

```
--- the beginning of file ---
[keyboard]
     type = 0;     # 0:106 1:101
     repeatw = 1; # 0:short 1:middle 2:long
     repeats = 1; # 0:fast 1:middle 2:slow

[mouse]
     speed = 1;    # 0:fast 1:middle 2:slow
     dblclk = 1;   # 0:fast 1:middle 2:slow
     lr = 1;       # 0:left hand 1:right hand

[atok]
     mode = 0;     # 0:roma 1:kana
     bind = 0;     # 0:setting1 1:setting2

[laststatus]
     softkb_onoff = 1;
      softkb_qwert = 1;
--- the end of file ---
```

conf/filetype.ini

Contains settings for each generic-data file type (currently used only for icon settings).

```
--- the beginning of file ---
[audio]
icon="/icon/audio/icon.sys"

[image]
icon="/icon/image/icon.sys"

[text]
icon="/icon/text/icon.sys"
```

```
[html]
icon="/icon/html/icon.sys"

[video]
icon="/icon/video/icon.sys"
--- the end of file ---
```

(Note: The library should be used to access these files.)

Common Partition (__common)

The common partition (__common) is used to store data common to all applications.

- Arbitrary data can be stored in the common partition. In addition to memory card data, this can include graphic, sound, video, text data, and the like (hereafter referred to as "generic data"). Note, however, that this partition is subject to the same restrictions as the application partition: program data cannot be stored here.
- The common partition has a directory structure.
- The common partition has an initial size of 1 GB and is expanded in increments of 1 GB. If the available free space in the common partition is not big enough for an application's requirements, the partition can be expanded freely. If, however, the hard disk drive's remaining capacity is insufficient and the partition cannot be expanded as much as intended, the user should be notified, and the application must continue running without crashing.
- Applications are prohibited from deleting the common partition.

The common partition currently has the following usage restrictions:

- Generic file and memory-card data cannot be stored in the root directory.
- All directories must be directly beneath the root directory. In other words, there can only be two layers in the directory hierarchy. These directories are referred to as "folders."
- The maximum number of folders is 256. Do not create folders in excess of this limit. Refer to the sample folder counting program to see how to count folders.
- The total number of files, directories, and memory card data in a single folder cannot exceed 1022 items. Refer to the sample data counting program to see how to count these items.
- The system sorts folders by their creation dates, and folder contents by their modification dates.

Folders

Currently, only the folder "Your Saves" (described next) can be used. The usual folder restrictions apply to this folder. (We plan to provide a specification allowing applications to create folders.)

- Folder name restrictions are as follows:
  character encoding: UTF8
  Length: 250 bytes (251 including NULL)
  Unusable characters: 11 ASCII characters: \ / : ; , * ? " < > |
  Naming conventions: none
- The lower 9 bits of the st_mode attribute must be set to the octal number 0777. All other attributes should be set to their defaults.

The "Your Saves" folder

The folder named "Your Saves" (with the exact character codes 0x59,0x6F,0x75,0x72,0x20,0x53,0x61,0x76,0x65,0x73) has a special meaning.

- It can be used as the default folder when accessing data in the common partition. If it doesn't exist, an application can create it.
- If the user renames or deletes it, a new "Your Saves" folder will be created.
- Apart from these limitations, this folder is similar to an ordinary folder, as described above.

Memory Card Data

- Basically, this conforms to the rules for creating saved data on a PS2 memory card.
- An application can only create PS2 saved data in a folder.
- Naming conventions for memory card data located in the common partition conform with those for the PS2 memory card.
- An application can copy, delete, move, or rename memory card data that it creates, but is absolutely prohibited from performing these operations on memory card data created by other applications.
- The memory-card data attribute st_attr (memory-card conversion attribute) and the st_mode attribute must be set as follows. Other attributes should be left at their defaults.

Table 3

Type	st_attr	st_mode (low 9 bits, in octal)
Directory (normal)	0xc4a7	0777
Directory (copy-protected)	0xc4af	0777
Included files	0x8497	0666

Note: The 0x4000 attribute has been recently added to the directory's str_attr. This bit indicates whether or not a file is memory card data. Always set this in the common partition.

Generic Data

- The use of generic data will be treated in a future document revision.

Temporary Partition (_tmp)

A partition identified as "_tmp" (note that unlike system partitions, the ID begins with a single underscore) can be used for temporary data storage. This can be used by applications installed on the hard disk drive, or even applications not installed. When using temporary partitions, you must observe the following conditions.

- If a _tmp partition already exists when an application is in use, first delete the existing one and create a new one.
- Given the above condition, it is important to remember to treat this partition as truly temporary, and that any files written to it will not survive the next program launching. Also, it must not be used by the save function of an application.
- The main partition that is initially created, and any subsequent sub-partitions should always be 1 GB or less (their total space can exceed 1 GB though).
- If insufficient space is available when creating a temporary partition, we recommend that the program should still be able to execute.
- No special message is displayed to the user when creating a _tmp partition, except in cases when the partition cannot be created or extended and the program cannot continue as a result. In this situation, the user should be notified. We also recommend a recovery procedure that will save the user's data.
- If the application performs the following functions, execute these functions after deleting the _tmp partition.
- power-off (shutdown) function.

Also note that the system will delete _tmp partitions while running.

Usable Range of UTF8 Character Codes (Glyphs)

For titles appearing in the icon.sys file, or deletion warning messages, characters within the JIS X 0208 character set that satisfy the following conditions can be used.

- Must be convertible to UCS2
- ASCII (0x20 - 0x7e)
- Non-kanji
- Level 1 kanji
- Level 2 kanji

The system includes a variable-width font and a fixed-width font. Which one to use can be determined by the character code. We recommend using variable-width fonts as much as possible.

Codes used by the variable-width font:

- ASCII (0x20 - 0x7e)
- Other Western characters and some symbols (reference table to be included in a future document revision).

(The fixed-width font is used for full-width alphanumeric characters beyond 0xff00 in UCS2. It is not used for half-width alphanumeric characters.)

Sample Programs

Folder Counting

```
      // Common partition mounted on "pfs0:".
      // Return value:
      //   1: New folder can be created
      //   0: Folder count already at maximum, cannot create a new folder
      //   <0: error code

        #define MAX_SEARCH_ENTRY    (1024)
        #define MAX_FOLDER          (256)
        int Judge_MakeNewFolder(void){
             static struct sce_dirent dirent;
             int folder_num = 0, ent, ret, fd;

             ret = sceDopen("pfs0:/");
             if(ret < 0){
                    return ret;  // Return error
             }

        fd = ret;

        for(ent = 0; ent < MAX_SEARCH_ENTRY; ent ++){

                    ret = sceDread(fd, &dirent);
                    if(ret <= 0){
                           break;  // No entries beyond this, or error
                    }

                    if(!strcmp(dirent.d_name, ".") ||      // Ignore "."
                       !strcmp(dirent.d_name, "..") ||           // Ignore
".."
                       !SCE_STM_ISDIR(dirent.d_stat.st_mode) ||
                                                      // Ignore non-
directories
                       dirent.d_stat.st_private[0] != 0xffff ||
                                               // Ignore uid other than
0xffff
                       dirent.d_stat.st_private[1] != 0xffff){
                                               // Ignore gid other than
0xffff
                       continue;
                    }

                    folder_num ++;  // Count OK

                    if(folder_num >= MAX_FOLDER){
                           break;  // Already exceeded maximum
                    }
             }

             {
                    int _ret = sceDclose(fd);

                    if(ret >= 0){
                          // If there was no error before close is
executed, then the close result is valid.
                          ret = _ret;
                    }
             }

             if(ret < 0){
                    return ret;
             }

             if(folder_num < MAX_FOLDER){
                    return 1;  // New folder can be created
             }
             else{
                    return 0;  // Cannot create more folders
             }
        }
```

Data Counting

```
        // Function
        //   path: folder's path (e.g. "pfs0:/folder_name")
        // Return value:
        //   1: New data can be created
        //   0: Data count already at maximum, cannot create new data
        //   <0: error code

         #define MAX_SEARCH_ENTRY    (1024)
         #define MAX_SAVEDATA        (1022)
         int Judge_MakeNewSavedata(const char *path){
         static struct sce_dirent dirent;
             int data_num = 0, ent, ret, fd;

             ret = sceDopen(path);
             if(ret < 0){
                    return ret;  // Return error
             }

             fd = ret;

             for(ent = 0; ent < MAX_SEARCH_ENTRY; ent ++){

                    ret = sceDread(fd, &dirent);
                    if(ret <= 0){
                          break;  // No entries beyond this, or error
                    }

                    if(!strcmp(dirent.d_name, ".") ||   // Ignore "."
                       !strcmp(dirent.d_name, "..") ||  // Ignore ".."

                       // Ignore directories that do not have the saved-data
attribute (0x4000)
                       (SCE_STM_ISDIR(dirent.d_stat.st_mode) &&
                       !(dirent.d_stat.st_attr&0x4000)) ||

                       // Ignore files with a ":" (resource files)
                       (strchr(dirent.d_name, ':') != NULL &&
                       SCE_STM_ISREG(dirent.d_stat.st_mode)) ||
                                // Ignore symbolic links
                                SCE_STM_ISLNK(dirent.d_stat.st_mode) ||

                                dirent.d_stat.st_private[0] != 0xffff ||
                                                        // Ignore uid other than
0xffff
                                dirent.d_stat.st_private[1] != 0xffff){
                                                        // Ignore gid other than
0xffff
                                continue;
                        }

                        data_num ++;  // Count OK
                }

                {
                        int _ret = sceDclose(fd);

                        if(ret >= 0){
                                // If there was no error before close is
executed, then the close result is valid.
                                ret = _ret;
                        }
                }

                if(ret < 0){
                        return ret;
                }

                if(data_num < MAX_SAVEDATA){
                        return 1;  // New data can be created
                }
                else{
                        return 0;  // Cannot create more data
                }
        }
```