libpgf Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Constant Definitions

SceFontFamilyCode

Font family codes

Definition

```
#include <font/libpgf.h>
enum SceFontFamilyCode {
        SCE_FONT_DEFAULT_FAMILY_CODE=(0),
        SCE_FONT_FAMILY_SANSERIF=(1),
        SCE_FONT_FAMILY_SERIF=(2),
        SCE_FONT_FAMILY_ROUNDED=(3)
};
```

Description

This is one piece of font design information.

It is the value that is returned in the *familyCode* member of the `SceFont_t_fontStyleInfo` type structure by the `sceFontGetFontList()` function for getting font information for PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita.

`SCE_FONT_FAMILY_SANSERIF`	Font of the sans serif family
`SCE_FONT_FAMILY_SERIF`	Font of the serif family
`SCE_FONT_FAMILY_ROUNDED`	Reserved value

If the following value is assigned to the *familyCode* member of the `SceFont_t_fontStyleInfo` type structure that is provided as an argument to the `sceFontFindOptimumFont()` or `sceFontFindFont()` function for finding a font, the font family that is the system default is selected.

`SCE_FONT_DEFAULT_FAMILY_CODE`	Default

See Also

`sceFontGetFontList()`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `SceFont_t_fontStyleInfo`

SceFontStyleCode

Style codes

Definition

```
#include <font/libpgf.h>
enum SceFontStyleCode {
        SCE_FONT_DEFAULT_STYLE_CODE=(0),
        SCE_FONT_STYLE_REGULAR=(1),
        SCE_FONT_STYLE_OBLIQUE=(2),
        SCE_FONT_STYLE_NARROW=(3),
        SCE_FONT_STYLE_NARROW_OBLIQUE=(4),
        SCE_FONT_STYLE_BOLD=(5),
        SCE_FONT_STYLE_BOLD_OBLIQUE=(6),
        SCE_FONT_STYLE_BLACK=(7),
        SCE_FONT_STYLE_BLACK_OBLIQUE=(8),
        SCE_FONT_STYLE_L=(101),
        SCE_FONT_STYLE_M=(102),
        SCE_FONT_STYLE_DB=(103),
        SCE_FONT_STYLE_B=(104),
        SCE_FONT_STYLE_EB=(105),
        SCE_FONT_STYLE_UB=(106)
};
```

Description

This is one piece of font design information.

It is the value that is returned in the *`style`* member variable of the `SceFont_t_fontStyleInfo` type structure by the `sceFontGetFontList()` function for getting font information for PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita.

The following style codes are mainly used for Latin fonts.

`SCE_FONT_STYLE_REGULAR`	Standard design
`SCE_FONT_STYLE_OBLIQUE`	Italic
`SCE_FONT_STYLE_NARROW`	Narrow
`SCE_FONT_STYLE_NARROW_OBLIQUE`	Narrow italic
`SCE_FONT_STYLE_BOLD`	Bold
`SCE_FONT_STYLE_BOLD_OBLIQUE`	Bold italic
`SCE_FONT_STYLE_BLACK`	Thicker bold
`SCE_FONT_STYLE_BLACK_OBLIQUE`	Thicker bold italic

The following style codes are mainly used for Japanese fonts.

`SCE_FONT_STYLE_L`	Narrower
`SCE_FONT_STYLE_M`	↑
`SCE_FONT_STYLE_DB`	
`SCE_FONT_STYLE_B`	
`SCE_FONT_STYLE_EB`	↓
`SCE_FONT_STYLE_UB`	Bolder

If the following value is assigned to the *`style`* member of the `SceFont_t_fontStyleInfo` type structure that is provided as an argument to the `sceFontFindOptimumFont()` or `sceFontFindFont()` function for finding a font, the style that is the system default is selected.

`SCE_FONT_DEFAULT_STYLE_CODE`	System standard

See Also

`sceFontGetFontList()`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `SceFont_t_fontStyleInfo`

SceFontImageBufferPixelFormatType

Pixel formats

Definition

```
#include <font/libpgf.h>
enum SceFontImageBufferPixelFormatType {
        SCE_FONT_USERIMAGE_DIRECT4_L=(0),
        SCE_FONT_USERIMAGE_DIRECT8=(2)
};
```

Description

This is the value that is assigned in the *pixelFormat* member variable of the SceFont_t_userImageBufferRec type structure that is passed to the sceFontGetCharGlyphImage() function for copying the font glyph images to the user memory space.

It specifies the format of the pixels that constitute a glyph image.

The sceFontGetCharGlyphImage() function copies the font glyph images to the memory that was specified by the user according to this format value.

SCE_FONT_USERIMAGE_DIRECT4_L	The low-order 4 bytes within 8 bits are the direct color grayscale 4 bits that are placed on the left side of the screen. The pixel value 0x0 means the minimum brightness and the pixel value 0xf means the maximum brightness.
SCE_FONT_USERIMAGE_DIRECT8	256-color direct color grayscale 8 bits. The pixel value 0x00 means minimum brightness and the pixel value 0xff means maximum brightness.

See Also

sceFontGetCharGlyphImage(),SceFont_t_userImageBufferRec

SceFontLanguageCode

Language codes (languages corresponding to the font)

Definition

```
#include <font/libpgf.h>
enum SceFontLanguageCode {
        SCE_FONT_DEFAULT_LANGUAGE_CODE=(0),
        SCE_FONT_LANGUAGE_J=(1),
        SCE_FONT_LANGUAGE_LATIN=(2),
        SCE_FONT_LANGUAGE_K=(3),
        SCE_FONT_LANGUAGE_CJK=(5)
};
```

Description

This is information about the language or languages corresponding to the font.

It is the value that is returned in the *languageCode* member variable of the `SceFont_t_fontStyleInfo` type structure by the `sceFontGetFontList()` function for getting font information for PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita.

`SCE_FONT_LANGUAGE_J`	Japanese
`SCE_FONT_LANGUAGE_LATIN`	English
`SCE_FONT_LANGUAGE_K`	Korean
`SCE_FONT_LANGUAGE_CJK`	Corresponds to Japanese and Korean

If the following value is assigned for the *languageCode* member of the `SceFont_t_fontStyleInfo` type structure that is provided as an argument to the `sceFontFindOptimumFont()` or `sceFontFindFont()` functions for finding a font, the font language code that is the system default is selected.

`SCE_FONT_DEFAULT_LANGUAGE_CODE`	Default

See Also

`sceFontGetFontList()`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `SceFont_t_fontStyleInfo`

SceFontRegionCode

Region codes (region corresponding to the font)

Definition

```
#include <font/libpgf.h>
enum SceFontRegionCode {
        SCE_FONT_GENERIC_REGION_CODE=(0),
        SCE_FONT_REGION_001=(1),
        SCE_FONT_REGION_002=(2),
        SCE_FONT_REGION_003=(3),
        SCE_FONT_REGION_004=(4),
        SCE_FONT_REGION_005=(5),
        SCE_FONT_REGION_006=(6),
        SCE_FONT_REGION_007=(7)
};
```

Description

This is information about the region corresponding to the font.

It is the value that is returned in the *regionCode* member variable of the `SceFont_t_fontStyleInfo` type structure by the `sceFontGetFontList()` function for getting font information for PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita.

`SCE_FONT_REGION_001`	Reserved value
`SCE_FONT_REGION_002`	Reserved value
`SCE_FONT_REGION_003`	Reserved value
`SCE_FONT_REGION_004`	Reserved value
`SCE_FONT_REGION_005`	Reserved value
`SCE_FONT_REGION_006`	Reserved value
`SCE_FONT_REGION_007`	Reserved value

If the following value is assigned for the *regionCode* member of the `SceFont_t_fontStyleInfo` type structure that is provided as an argument to the `sceFontFindOptimumFont()` or `sceFontFindFont()` functions for finding a font, a font for which the Region is unspecified is selected.

`SCE_FONT_GENERIC_REGION_CODE`	Unspecified

See Also

`sceFontGetFontList()`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `SceFont_t_fontStyleInfo`

SceFontFontVendorCountryCode

Font vendor country codes

Definition

```
#include <font/libpgf.h>
enum SceFontFontVendorCountryCode {
        SCE_FONT_GENERIC_COUNTRY_CODE=(0),
        SCE_FONT_COUNTRY_JAPAN=(1),
        SCE_FONT_COUNTRY_USA=(2),
        SCE_FONT_COUNTRY_KOREA=(3)
};
```

Description

This is the country code of the font vendor.

It is the value that is returned in the *countryCode* member variable of the `SceFont_t_fontStyleInfo` type structure by the `sceFontGetFontList()` function for getting font information for PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita.

`SCE_FONT_COUNTRY_JAPAN`	Japan
`SCE_FONT_COUNTRY_USA`	United States of America
`SCE_FONT_COUNTRY_KOREA`	Korea

If the following value is assigned for the *countryCode* member of the `SceFont_t_fontStyleInfo` type structure that is provided as an argument to the `sceFontFindOptimumFont()` or `sceFontFindFont()` functions for finding a font, a font for which the vendor's country is unspecified is selected.

`SCE_FONT_GENERIC_COUNTRY_CODE`	Country unspecified

See Also

`sceFontGetFontList()`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `SceFont_t_fontStyleInfo`

SceFontBoolValue

Boolean values

Definition

```
#include <font/libpgf.h>
enum SceFontBoolValue {
        SCE_FONT_FALSE=(0),
        SCE_FONT_TRUE=(1)
};
```

Description

This is used to specify the Boolean values that are handled by libpgf.

SCE_FONT_FALSE	False
SCE_FONT_TRUE	True

SceFontDataAccessMode

Access modes

Definition

```
#include <font/libpgf.h>
enum SceFontDataAccessMode {
        SCE_FONT_FILEBASEDSTREAM=(0),
        SCE_FONT_MEMORYBASEDSTREAM=(1)
};
```

Description

This value is used to specify the mode for accessing font data.

`SCE_FONT_FILEBASEDSTREAM`	Font data in a file is handled directly as a file.
`SCE_FONT_MEMORYBASEDSTREAM`	All font data in a file is read into memory and handled as data in memory. The file itself is closed when the data is read into memory. Memory that was allocated for file reading by using the `sceFontOpen()` or `sceFontOpenUserFile()` function is released when the `sceFontClose()` function is executed.

See Also

`sceFontOpen()`, `sceFontOpenUserFile()`, `sceFontOpenUserMemory()`

SCE_FONT_SUBSTYLE_xxx

Substyle attribute mask value (mainly raster processing bold and slant specifications)

Definition

```
#include <font/libpgf.h>
#define SCE_FONT_SUBSTYLE_VERTICALLAYOUT (0x0001)
#define SCE_FONT_SUBSTYLE_PSEUDO_BOLD (0x0002)
#define SCE_FONT_SUBSTYLE_PSEUDO_SLANT (0x0004)
```

Description

This is one piece of font design information.

It is the value that is returned in the *subStyle* member variable of the SceFont_t_fontStyleInfo type structure by the sceFontGetFontList() function for getting font information for PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita.

SCE_FONT_SUBSTYLE_VERTICALLAYOUT	Typeface for vertical layout
SCE_FONT_SUBSTYLE_PSEUDO_BOLD	Style for which pseudo boldface processing was executed
SCE_FONT_SUBSTYLE_PSEUDO_SLANT	Style for which pseudo italic processing was executed

(*) The PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita do not have the substyles described above.

See Also

sceFontGetFontList(),sceFontFindOptimumFont(),sceFontFindFont(), SceFont_t_fontStyleInfo

SCE_FONT_FONTNAME_LENGTH

Font name maximum length

Definition

```
#include <font/libpgf.h>
#define SCE_FONT_FONTNAME_LENGTH (64)
```

Description

This is the maximum length of string data that is used when libpgf handles a font name.
The font name is the name of the font, not the filename of the font.
Different font files may have the same font name.

See Also

```
sceFontGetFontList(),sceFontFindOptimumFont(),sceFontFindFont(),
SceFont_t_fontStyleInfo
```

SCE_FONT_FONTFILENAME_LENGTH

Font filename maximum length

Definition

```
#include <font/libpgf.h>
#define SCE_FONT_FONTFILENAME_LENGTH (64)
```

Description

This is the maximum length of string data that includes a device name and directory name, which is used when libpgf handles a font filename.

See Also

`sceFontGetFontList()`,`sceFontFindOptimumFont()`,`sceFontFindFont()`,
`SceFont_t_fontStyleInfo`

SCE_FONT_MAX_OPEN

Maximum number of fonts that can be open simultaneously

Definition

```
#include <font/libpgf.h>
#define SCE_FONT_MAX_OPEN (9)
```

Description

This is the maximum number of fonts that can be open simultaneously, which has been determined within libpgf.

When multiple library instances of libpgf are generated, the maximum number of fonts that can be open simultaneously per library instance is less than the `SCE_FONT_MAX_OPEN` value. The upper bound of the file descriptor resources assigned by the kernel for accessing the filesystem is related to this.

See Also

`sceFontNewLib()`, `sceFontOpen()`, `sceFontClose()`

Variable Types

List of Variable Types

Simple variable types defined in libpgf.h

Definition

Type Name	Entity	Use
`SceFont_t_u64`	`unsigned long long`	Unsigned 64-bit type
`SceFont_t_s64`	`signed long long`	Signed 64-bit type
`SceFont_t_u32`	`unsigned long`	Unsigned 32-bit type
`SceFont_t_s32`	`signed long`	Signed 32-bit type
`SceFont_t_u16`	`unsigned short`	Unsigned 16-bit type
`SceFont_t_s16`	`signed short`	Signed 16-bit type
`SceFont_t_u8`	`unsigned char`	Unsigned 8-bit type
`SceFont_t_s8`	`signed char`	Signed 8-bit type
`SceFont_t_f32`	`float`	32-bit floating-point type
`SceFont_t_f64`	`double`	64-bit floating-point type
`SceFont_t_bool`	`SceFont_t_u32`	Boolean type
`SceFont_t_libId`	`void *`	`sceFont` library handle pointer
`SceFont_t_fontId`	`void *`	`sceFont` font handle
`SceFont_t_pointer`	`void *`	General pointer type
`SceFont_t_handle`	`void *`	General ID type (same meaning as `SceFont_t_pointer`)
`SceFont_t_error`	`SceFont_t_s32`	For error values
`SceFont_t_int`	`SceFont_t_s32`	Integer type
`SceFont_t_charCode`	`SceFont_t_u16`	Character code type
`SceFont_t_string`	`SceFont_t_charCode *`	String type
`SceFont_t_fontIndex`	`SceFont_t_s32`	Font number determined by system

Datatypes

SceFont_t_irect

General rectangle data type

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_irect {
        SceFont_t_u16 width;
        SceFont_t_u16 height;
} SceFont_t_irect;
```

Members

width	Width represented by a 16-bit integer
height	Height represented by a 16-bit integer

Description

This is a data type (structure) for handling the width and height of a general rectangle.

It is used by data types or functions that are used for processing in which the units are the pixels of a glyph image.

See Also

`SceFont_t_userImageBufferRec`, `sceFontGetCharImageRect()`

SceFont_t_rect

General rectangle data type

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_rect {
        SceFont_t_u32 width;
        SceFont_t_u32 height;
} SceFont_t_rect;
```

Members

width	Width represented by a 32-bit integer
height	Height represented by a 32-bit integer

Description

This is a data type (structure) for handling the width and height of a general rectangle.

It is used by data types or functions that are used for processing in which the units are the pixels of a glyph image.

SceFont_t_cacheSystemInterface

Data type of interface with font cache system

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_cacheSystemInterface {
        SceFont_t_pointer *cacheInstance;
        SceFont_t_s32 (*lockFunc)
              (
              SceFont_t_pointer
              );
        SceFont_t_s32 (*unlockFunc)
              (
              SceFont_t_pointer
              );
        SceFont_t_pointer (*findFunc)
              (
              SceFont_t_pointer,
              SceFont_t_u32,
              SceFont_t_pointer,
              SceFont_t_bool *
              );
        SceFont_t_s32 (*writeKeyValueToCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer
              );
        SceFont_t_s32 (*write0ToCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_int
              );
        SceFont_t_s32 (*write1ToCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_int
              );
        SceFont_t_s32 (*write2ToCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_int
              );
        SceFont_t_s32 (*write3ToCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_int
              );
        SceFont_t_s32 (*read0FromCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer
              );
        SceFont_t_s32 (*read1FromCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer
              );
        SceFont_t_s32 (*read2FromCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer
              );
        SceFont_t_s32 (*read3FromCacheFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_pointer
              );
} SceFont_t_cacheSystemInterface;
```

Members

cacheInstance	Value passed to first argument of cache interface function group
lockFunc	Pointer to function for locking the cache
unlockFunc	Pointer to function for unlocking the cache
findFunc	Pointer to function for checking whether or not the cache exists
writeKeyValueToCacheFunc	Pointer to function for writing a key value to the cache
write0ToCacheFunc	Pointer to function for writing data0 to the cache
write1ToCacheFunc	Pointer to function for writing data1 to the cache
write2ToCacheFunc	Pointer to function for writing data2 to the cache
write3ToCacheFunc	Pointer to function for writing data3 to the cache
read0FromCacheFunc	Pointer to function for reading data0 from the cache
read1FromCacheFunc	Pointer to function for reading data1 from the cache
read2FromCacheFunc	Pointer to function for reading data2 from the cache
read3FromCacheFunc	Pointer to function for reading data3 from the cache

Description

libpgf can be assigned a cache system from an outside source.

libpgf communicates with this cache system via the function group assigned by `SceFont_t_cacheSystemInterface`. When libpgf calls this function group, it passes *cacheInstance* in the first argument.

The cache system function group specifications expected by libpgf are as follows.

Cache system initialization and termination processing are not performed from within libpgf. Make sure that the application performs processing as necessary before libpgf initialization and after libpgf termination processing.

result = lockFunc (cacheInstance)

This function locks the cache.

The cache system must not accept requests from elsewhere while the cache is locked.

When locking is successful, 0 is returned for *result*. When locking fails, -1 is returned.

result = unlockFunc (cacheInstance)

This function unlocks the cache.

When unlocking is successful, 0 is returned for `result`. When unlocking fails, -1 is returned.

cacheSlot = findFunc (cacheInstance, hashValue, key, result)

This function uses the value of *`hashValue`* to check whether or not the data indicated by *`key`* exists within the cache.

`hashValue` is a hash value generated by libpgf. If this value is used, comparison processing with data in the cache can be performed quickly. However, libpgf does not expect that the cache system always implements and uses this value. The cache system itself is permitted to generate a hash value from *`key`*.

`key` is a pointer to a `SceFontCacheKey` type variable (structure), and `SceFontCacheKey` has four member variables. This function compares data within the cache with *`key`* and if all four member variables match, the cache data and key are considered to be the same.

When the same data is not found in the cache, this function writes `SCE_FONT_FALSE` in `*`*`result`* and returns NULL in *`cacheSlot`*.

When the same data is found in the cache, this function writes `SCE_FONT_TRUE` in `*`*`result`* and returns a pointer to the cache slot where it was found in *`cacheSlot`*.

result = writeKeyValueToCacheFunc (cacheInstance, cacheSlot, key)

This function stores the value of *`key`* in the specified cache slot *`cacheSlot`*.

If processing fails for some reason, -1 is returned for `result`. Otherwise, 0 is returned for `result`.

result = write0ToCacheFunc (cacheInstance, cacheSlot, data, dataSize)

This function stores data with a size of the number of bytes indicated by *`dataSize`* from the area indicated by the pointer *`data`* as one data of the cache slot specified by *`cacheSlot`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function stores data in one (the 0th one) of those data blocks.

If processing fails for some reason, -1 is returned for `result`. Otherwise, 0 is returned for `result`.

result = write1ToCacheFunc (cacheInstance, cacheSlot, data, dataSize)

This function stores data with a size of the number of bytes indicated by dataSize from the area indicated by the pointer *`data`* as one data of the cache slot specified by *`cacheSlot`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function stores data in one (the 1st one) of those data blocks.

If processing fails for some reason, -1 is returned for `result`. Otherwise, 0 is returned for `result`.

result = write2ToCacheFunc (cacheInstance, cacheSlot, data, dataSize)

This function stores data with a size of the number of bytes indicated by *`dataSize`* from the area indicated by the pointer *`data`* as one data of the cache slot specified by *`cacheSlot`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function stores data in one (the 2nd one) of those data blocks.

If processing fails for some reason, -1 is returned for result. Otherwise, 0 is returned for result.

result = write3ToCacheFunc (cacheInstance, cacheSlot, data, dataSize)

This function stores data with a size of the number of bytes indicated by dataSize from the area indicated by the pointer data as one data of the cache slot specified by *`cacheSlot`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function stores data in one (the 3rd one) of those data blocks.

If processing fails for some reason, -1 is returned for *`result`*. Otherwise, 0 is returned for *`result`*.

result = read0FromCacheFunc (cacheInstance, cacheSlot, dst)

This function copies one (the 0th one) of the data from the cache slot specified by *`cacheSlot`* to the area indicated by the pointer *`dst`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function copies the data that was stored in one (the 0th one) of those data blocks to *`dst`*.

If processing fails for some reason, -1 is returned for *`result`*. Otherwise, 0 is returned for *`result`*.

result = read1FromCacheFunc (cacheInstance, cacheSlot, dst)

This function copies one (the 1st one) of the data from the cache slot specified by *`cacheSlot`* to the area indicated by the pointer *`dst`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function copies the data that was stored in one (the 1st one) of those data blocks to *`dst`*.

If processing fails for some reason, -1 is returned for *`result`*. Otherwise, 0 is returned for *`result`*.

result = read2FromCacheFunc (cacheInstance, cacheSlot, dst)

This function copies one (the 2nd one) of the data from the cache slot specified by *`cacheSlot`* to the area indicated by the pointer *`dst`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function copies the data that was stored in one (the 2nd one) of those data blocks to *`dst`*.

If processing fails for some reason, -1 is returned for *`result`*. Otherwise, 0 is returned for *`result`*.

result = read3FromCacheFunc (cacheInstance, cacheSlot, dst)

This function copies one (the 3rd one) of the data from the cache slot specified by *`cacheSlot`* to the area indicated by the pointer *`dst`*.

This *`cacheSlot`* is the value that was returned as the return value by *`findFunc`*.

libpgf expects the cache system to manage four data blocks in one cache slot.

This function copies the data that was stored in one (the 3rd one) of those data blocks to *`dst`*.

If processing fails for some reason, -1 is returned for *`result`*. Otherwise, 0 is returned for *`result`*.

libpgf operates even if no cache system is assigned. If NULL is assigned for the member variable *`cache`* of the `SceFont_t_initRec` structure that is assigned by an argument in the `sceFontNewLib()` function, libpgf operates with no cache system.

When a font is accessed in `SCE_FONT_MEMORYBASEDSTREAM` mode, the use of `SceFont_t_cacheSystemInterface` does not contribute to an increase in processing speed, so the cache mechanism is not used. The cache mechanism is only used when the font is accessed in `SCE_FONT_FILEBASEDSTREAM` mode.

See Also

`sceFontNewLib()`, `SceFont_t_initRec`, sample program "fontcache"

`SCE_FONT_MEMORYBASEDSTREAM`, `SCE_FONT_FILEBASEDSTREAM`

SceFont_t_initRec

Data type of information specified when use of libpgf begins

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_initRec {
        SceFont_t_pointer userData;
        SceFont_t_u32 maxNumFonts;
        SceFont_t_cacheSystemInterface *cache;
        SceFont_t_pointer (*allocFunc)
              (
              SceFont_t_pointer,
              SceFont_t_u32
              );
        void (*freeFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer
              );
        SceFont_t_handle (*openFunc)
              (
              SceFont_t_pointer,
              SceFont_t_pointer,
              SceFont_t_error *
              );
        SceFont_t_error (*closeFunc)
              (
              SceFont_t_pointer,
              SceFont_t_handle
              );
        SceFont_t_u32 (*readFunc)
              (
              SceFont_t_pointer,
              SceFont_t_handle,
              SceFont_t_pointer,
              SceFont_t_u32,
              SceFont_t_u32,
              SceFont_t_error *
              );
        SceFont_t_error (*seekFunc)
              (
              SceFont_t_pointer,
              SceFont_t_handle,
              SceFont_t_u32
              );
        SceFont_t_s32 (*onErrorFunc)
              (
              SceFont_t_pointer,
              SceFont_t_s32
              );
        SceFont_t_s32 (*whenDoneReadFunc)
              (
              SceFont_t_pointer,
              SceFont_t_s32
              );
} SceFont_t_initRec;
```

Members

userData	Pointer to user data
maxNumFonts	Maximum number of fonts that are open simultaneously
cache	Cache system instance handle
allocFunc	Memory allocation function
freeFunc	Memory release function
openFunc	File open (read-only) function
closeFunc	File close function
readFunc	File read function
seekFunc	File seek function
onErrorFunc	Reserved area
whenDoneReadFunc	Reserved area

Description

For *userData*, specify the value that is passed in the first argument when the user-provided memory allocation and release functions or file access functions are called by libpgf.

For *maxNumfonts*, specify the number of fonts that can be open simultaneously. The maximum value that can be specified is `SCE_FONT_MAX_OPEN`. This value cannot be exceeded.

For *cache*, specify a user-provided cache system instance.

The memory allocation and release function specifications expected by libpgf are as follows.

p = allocFunc (userData, size)

userData is the value that was assigned for the member variable *userData* of the `SceFont_t_initRec` structure.

This function allocates memory with a size of *size* bytes (0 <= *size*) aligned to a 4-byte boundary and returns p. When memory allocation fails, NULL is returned.

freeFunc (userData, p)

userData is the value that was assigned for the member variable *userData* of the `SceFont_t_initRec` structure.

This function releases the memory of the area indicated by *p*.

When a font was opened in `SCE_FONT_FILEBASEDSTREAM` mode by using `sceFontOpenUserFile()`, libpgf expects that file I/O for accessing that file is assigned from outside.

The file I/O function group specifications expected by libpgf are as follows.

filehandle = openFunc (userData, filename, errorCode)

userData is the value that was assigned for the member variable *userData* of the `SceFont_t_initRec` structure.

This function opens the file having the filename indicated by *filename*. It returns the file handle value that was returned by the last file open function in *filehandle*.

If processing for opening the file is successful, `SCE_OK` is stored in **errorCode*. If this processing fails, `SCE_FONT_ERROR_FILEOPEN` is stored.

errorCode = closeFunc (userData, filehandle)

userData is the value that was assigned for the member variable *userData* of the `SceFont_t_initRec` structure.

This function closes the open file that was assigned by *filehandle*.

If processing for closing the file is successful, `SCE_OK` is returned in *errorCode*. If this processing fails, `SCE_FONT_ERROR_FILECLOSE` is returned.

readCount = readFunc (userData, filehandle, p, size, num, errorCode)

userData is the value that was assigned for the member variable *userData* of the SceFont_t_initRec structure.

This function reads *num* units of data to the area indicated by *p* with a size of *size* bytes from the open file assigned by *filehandle* and returns the number of units that were actually read in *readCount*.

If processing for reading the file is successful, SCE_OK is stored in **errorCode*. If this processing fails, SCE_FONT_ERROR_FILEREAD is stored, and 0 is returned in *readCount*.

errorCode = seekFunc (userData, filehandle, offset)

userData is the value that was assigned for the member variable *userData* of the SceFont_t_initRec structure.

This function performs a seek operation up to the position indicated by offset within the open file assigned by filehandle.

If file seek processing is successful, SCE_OK is returned in *errorCode*. If this processing fails, SCE_FONT_ERROR_FILESEEK is returned.

libpgf assigns *userData* for the first argument of all function calls indicated by SceFont_t_initRec.

See Also

SceFont_t_cacheSystemInterface, sceFontNewLib()

SceFont_t_fontStyleInfo

Data type used for getting installed font information and for finding fonts

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_fontStyleInfo {
        SceFont_t_f32 hSize;
        SceFont_t_f32 vSize;
        SceFont_t_f32 hResolution;
        SceFont_t_f32 vResolution;
        SceFont_t_f32 weight;
        SceFont_t_u16 familyCode;
        SceFont_t_u16 style;
        SceFont_t_u16 subStyle;
        SceFont_t_u16 languageCode;
        SceFont_t_u16 regionCode;
        SceFont_t_u16 countryCode;
        SceFont_t_u8 fontName [SCE_FONT_FONTNAME_LENGTH];
        SceFont_t_u8 fileName [SCE_FONT_FONTFILENAME_LENGTH];
        SceFont_t_u32 extraAttributes;
        SceFont_t_u32 expireDate;
} SceFont_t_fontStyleInfo;
```

Members

hSize	Number of horizontal points
vSize	Number of vertical points
hResolution	Resolution in horizontal direction assumed by font (dpi value)
vResolution	Resolution in vertical direction assumed by font (dpi value)
weight	Weight value
familyCode	Family code
style	Style
subStyle	Substyle
languageCode	Language code
regionCode(*)	Region code
countryCode	Font vendor country code
fontName	Font name string
fileName	Font filename string
extraAttributes	Additional attribute information
expireDate(*)	Expiration date

Description

This is a structure that is handled by the `sceFontGetFontList()` function for getting font information about PSP™-compatible grayscale dot fonts that are installed on the PlayStation®Vita.

Before an application program that uses libpgf uses `sceFontOpen()` to open a libpgf font, it should use `sceFontGetFontList()` to check the fonts that are actually installed on the PlayStation®Vita to determine an appropriate font or it should use `sceFontFindOptimumFont()` to determine the font that should be selected.

One point is 1/72 inch. Also, dpi is an abbreviation of dots per inch.

(*) There are no PSP™-compatible grayscale dot fonts installed on the PlayStation®Vita for which a Region code is specified or for which an expiration date is specified.

See Also

`SceFontFamilyCode`, `SceFontStyleCode`, `SceFontLanguageCode`, `SceFontRegionCode`, `SceFontFontVendorCountryCode`, `SCE_FONT_SUBSTYLE_xxx`, `SCE_FONT_FONTNAME_LENGTH`, `SCE_FONT_FONTFILENAME_LENGTH`, `SceFont_t_fontInfo`, `sceFontGetFontList()`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `sceFontGetFontInfoByIndexNumber()`

SceFont_t_userImageBufferRec

Data type used when libpgf copies glyph images to user memory area

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_userImageBufferRec {
        SceFont_t_u32 pixelFormat;
        SceFont_t_s32 xPos64;
        SceFont_t_s32 yPos64;
        SceFont_t_irect rect;
        SceFont_t_u16 bytesPerLine;
        SceFont_t_u16 reserved;
        SceFont_t_u8 *buffer;
} SceFont_t_userImageBufferRec;
```

Members

pixelFormat	Pixel format
xPos64	X-position of reference point for writing
yPos64	Y-position of reference point for writing
rect	Buffer horizontal and vertical size Specifies the horizontal and vertical size of the image `buffer`. The units of this value are pixels
bytesPerLine	Number of bytes per horizontal line of `buffer` Specifies the number of bytes per horizontal line of the image `buffer`.
reserved	Padding (always set 0)
buffer	Pointer to `buffer` area Specifies the address of memory allocated by the user application. Be sure to specify an area that was allocated with a size of at least `bytesPerLine * rect.height` bytes.

Specify the format of one pixel for `pixelFormat`. The following `SCE_FONT_USERIMAGE_xxx` values can be specified.

`SCE_FONT_USERIMAGE_DIRECT4_L`	This is a pixel format in which 2 pixels are contained in 1 byte. The high-order 4 bits are assumed to be placed at the left side of the screen and the low-order four bits are assumed to be placed at the right side of the screen. The value 0x0 means the minimum brightness, and the value 0xf means the maximum brightness.
`SCE_FONT_USERIMAGE_DIRECT8`	This is a pixel format in which 1 pixel is contained in 1 byte. The value 0x00 means the minimum brightness and the pixel value 0xff means the maximum brightness.

For `xPos64` and `yPos64`, specify the position on the character base line that is to be the starting point when the `sceFontGetCharGlyphImage()` or `sceFontGetCharGlyphImage_Clip()` function copies a character. The units of this value are 1/64 of a pixel. libpgf, which also processes numeric values less than 1 pixel, calculates the position and brightness for copying the glyph image of a character.

Description

libpgf copies the glyph image of a character to the user memory space according to the `sceFontGetCharGlyphImage()` or `sceFontGetCharGlyphImage_Clip()` function.

The destination memory to which both of these functions copy the glyph image is determined according to `SceFont_t_userImageBufferRec` type data.

By setting an appropriate CLUT, the user application can handle *`buffer`* as a texture and display it by mapping it to a polygon.

See Also

`SceFontImageBufferPixelFormatType`, `sceFontGetCharGlyphImage()`, `sceFontGetCharGlyphImage_Clip()`

SceFont_t_iGlyphMetricsInfo

Data type of character glyph metrics information (fixed-point format)

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_iGlyphMetricsInfo {
        SceFont_t_u32 width64;
        SceFont_t_u32 height64;
        SceFont_t_s32 ascender64;
        SceFont_t_s32 descender64;
        SceFont_t_s32 horizontalBearingX64;
        SceFont_t_s32 horizontalBearingY64;
        SceFont_t_s32 verticalBearingX64;
        SceFont_t_s32 verticalBearingY64;
        SceFont_t_s32 horizontalAdvance64;
        SceFont_t_s32 verticalAdvance64;
} SceFont_t_iGlyphMetricsInfo;
```

Members

width64	Indicates the character width. The units are 1/64 pixel.
height64	Indicates the character height. The units are 1/64 pixel.
ascender64	Indicates the character ascender. The units are 1/64 pixel. Normally, this is the same value as *horizontalBearingY64*.
descender64	Indicates the character *descender*. The units are 1/64 pixel. Normally, this is the same value as *horizontalBearingY64* - *height64*.
horizontalBearingX64	This is the bearing value in the X-axis direction for horizontal character layout. The units are 1/64 pixel.
horizontalBearingY64	This is the bearing value in the Y-axis direction for horizontal character layout. The units are 1/64 pixel.
verticalBearingX64 (*)	This is the bearing value in the X-axis direction for vertical character layout. The units are 1/64 pixel.
verticalBearingY64 (*)	This is the bearing value in the Y-axis direction for vertical character layout. The units are 1/64 pixel.
horizontalAdvance64	This is the advance in the X-axis direction for horizontal character layout. The units are 1/64 pixel.
verticalAdvance64 (*)	This is the advance in the Y-axis direction for vertical character layout. The units are 1/64 pixel.

(*) Although all characters in the fonts that are installed on the PlayStation®Vita have numeric values for vertical layout metrics, vertical layout is not guaranteed.

Description

This is a data type for representing glyph metrics information for one character.

It is used by the `sceFontGetFontInfo()` function for indicating the maximum values for glyph metrics of all characters that are included in a font as typical values for the entire font.

See Also

`SceFont_t_fGlyphMetricsInfo`, `SceFont_t_charInfo`, `SceFont_t_fGlyphMetricsInfo`, `sceFontGetFontInfo()`, `sceFontGetCharInfo()`

SceFont_t_charInfo

Data type of character-specific information

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_charInfo {
        SceFont_t_u32 bitmapWidth;
        SceFont_t_u32 bitmapHeight;
        SceFont_t_s32 bitmapLeft;
        SceFont_t_s32 bitmapTop;
        SceFont_t_iGlyphMetricsInfo glyphMetrics;
        SceFont_t_u8 reserved [4];
} SceFont_t_charInfo;
```

Members

bitmapWidth	Bitmap width Indicates the horizontal size of the glyph image. The units are pixels.
bitmapHeight	Bitmap height Indicates the vertical size of the glyph image. The units are pixels.
bitmapLeft	Horizontal position of baseline origin in bitmap Indicates the value of the X-axis direction offset from the origin of the left edge of the glyph image. The units are pixels.
bitmapTop	Vertical position of baseline origin in bitmap Indicates the value of the Y-axis direction offset from the origin of the top edge of the glyph image. The units are pixels.
glyphMetrics	Character metrics information Indicates glyph metrics information.
reserved	Undefined

Description

This is a data type for indicating the numeric values of the bitmap size of the glyph image of a certain character and the offset from the baseline origin of that image and character metrics information.

See Also

`SceFont_t_iGlyphMetricsInfo`, `sceFontGetCharInfo()`

SceFont_t_fGlyphMetricsInfo

Data type of character glyph metrics information (floating-point format)

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_fGlyphMetricsInfo {
        SceFont_t_f32 width;
        SceFont_t_f32 height;
        SceFont_t_f32 ascender;
        SceFont_t_f32 descender;
        SceFont_t_f32 horizontalBearingX;
        SceFont_t_f32 horizontalBearingY;
        SceFont_t_f32 verticalBearingX;
        SceFont_t_f32 verticalBearingY;
        SceFont_t_f32 horizontalAdvance;
        SceFont_t_f32 verticalAdvance;
} SceFont_t_fGlyphMetricsInfo;
```

Members

width	Indicates the character width. The units are pixels.
height	Indicates the character height. The units are pixels.
ascender	Indicates the character ascender. The units are pixels. Normally, this is the same value as *horizontalBearingY64*.
descender	Indicates the character *descender*. The units are pixels. Normally, this is the same value as *horizontalBearingY64* - *height64*.
horizontalBearingX	This is the bearing value in the X-axis direction for horizontal character layout. The units are pixels.
horizontalBearingY	This is the bearing value in the Y-axis direction for horizontal character layout. The units are pixels.
verticalBearingX (*)	This is the bearing value in the X-axis direction for vertical character layout. The units are pixels.
verticalBearingY (*)	This is the bearing value in the Y-axis direction for vertical character layout. The units are pixels.
horizontalAdvance	This is the advance in the X-axis direction for horizontal character layout. The units are pixels.
verticalAdvance (*)	This is the advance in the Y-axis direction for vertical character layout. The units are pixels.

(*) Although all characters in the fonts that are installed on the PlayStation®Vita have numeric values for vertical layout metrics, vertical layout is not guaranteed.

Description

This is a data type for representing glyph metrics information for one character.

It is used by the `sceFontGetFontInfo()` function for indicating the maximum values for glyph metrics of all characters that are included in a font as typical values for the entire font.

See Also

`SceFont_t_iGlyphMetricsInfo`, `SceFont_t_charInfo`, `sceFontGetFontInfo()`, `sceFontGetCharInfo()`

SceFont_t_fontInfo

Data type of information related to font data in general

Definition

```
#include <font/libpgf.h>
typedef struct SceFont_t_fontInfo {
        SceFont_t_iGlyphMetricsInfo maxIGlyphMetrics;
        SceFont_t_fGlyphMetricsInfo maxFGlyphMetrics;
        SceFont_t_u16 maxGlyphBitmapWidth;
        SceFont_t_u16 maxGlyphBitmapHeight;
        SceFont_t_u32 numChars;
        SceFont_t_u32 numSubChars;
        SceFont_t_fontStyleInfo fontStyleInfo;
        SceFont_t_u8 pixelDepth;
        SceFont_t_u8 reserved [3];
} SceFont_t_fontInfo;
```

Members

maxIGlyphMetrics	Maximum metrics value (fixed-point representation) This is the maximum value of metrics information of all characters included in the font data represented as a fixed-point numeric value. The units of the member variable are 1/64 pixel.
maxFGlyphMetrics	Maximum metrics value (floating-point representation) This is the maximum value of metrics information of all characters included in the font data represented as a floating-point numeric value. The units of the member variable are pixels.
maxGlyphBitmapWidth	Maximum bitmap width Indicates the maximum horizontal size of the bitmaps of all characters included in the font data. The units are pixels
maxGlyphBitmapHeight	Maximum bitmap height Indicates the maximum vertical size of the bitmaps of all characters included in the font data. The units are pixels
numChars	Number of recorded character types Indicates the number of all character types included in the font data.
numSubChars	Reserved area (always 0)
fontStyleInfo	Indicates font style information.
pixelDepth	Number of bits per pixel of a glyph image Indicates the number of bits per pixel constituting the glyph image of a character included in the font data.
reserved	Undefined

Description

This is a data type for handling information related to one font in general.

maxIGlyphMetrics and *maxFGlyphMetrics* indicate maximum values over a typical range for a given character set. Note that a font may contain characters with glyphs that exceed these maximum values. An example is the character Å (U+212B:ANGSTROM SIGN). *maxIGlyphMetrics* and *maxFGlyphMetrics* contained in SceFont_t_fontInfo should be handled as target values for laying out glyphs in the given font (such as for line spacing). Accurate values for individual characters can be obtained from the SceFont_t_charInfo structure via the sceFontGetCharInfo() function.

See Also

sceFontGetFontInfo(), SceFont_t_iGlyphMetricsInfo, SceFont_t_fGlyphMetricsInfo, SceFont_t_fontStyleInfo

SceFontCacheKey

Data type used as key in font cache system

Definition

```
#include <font/libpgf.h>
typedef struct SceFontCacheKey {
        int keyValue0;
        int keyValue1;
        int keyValue2;
        int keyValue3;
} SceFontCacheKey;
```

Members

keyValue0	Comparison key 0
keyValue1	Comparison key 1
keyValue2	Comparison key 2
keyValue3	Comparison key 3

Description

This is a data type assigned by libpgf as comparison keys in the font cache system.

When these four comparison keys are all the same value, libpgf has the font cache system behave so that they are handled as the same data.

See Also

`SceFont_t_cacheSystemInterface`, `SceFont_t_initRec`, `sceFontNewLib()`

Functions

sceFontNewLib

Generate libpgf library instance

Definition

```
#include <font/libpgf.h>
SceFont_t_libId libID = sceFontNewLib (
        SceFont_t_initRec *initParam,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

initParam	Parameters such as pointer to callback function to be used
errorCode	Address for storing error code

Return Values

If the function completes normally, a pointer to the library instance is returned in *libID*.

If an error occurs, NULL is returned in *libID*, and either SCE_OK, SCE_FONT_ERROR_NOMEMORY, or SCE_FONT_ERROR_ARG is stored in **errorCode*.

Description

This function generates a libpgf library instance.

Multiple library instances can be generated at the same time.

Examples

```
static SceFont_t_pointer cb_Alloc (
        SceFont_t_pointer pMyData,
        SceFont_t_u32 size
 );
static void cb_Free (
        SceFont_t_pointer pMyData,
        SceFont_t_pointer p
);
static SceFont_t_handle cb_Open (
        SceFont_t_pointer pMyData,
        SceFont_t_pointer pFilename,
        SceFont_t_error *errorCode
);
static SceFont_t_error cb_Close (
        SceFont_t_pointer pMyData,
        SceFont_t_handle FileID
);
static SceFont_t_u32 cb_Read (
        SceFont_t_pointer pMyData,
        SceFont_t_handle FileID,
        SceFont_t_pointer pBuffer,
        SceFont_t_u32 NumByte,
        SceFont_t_u32 NumUnit,
        SceFont_t_error *errorCode
);
static SceFont_t_error cb_Seek (
        SceFont_t_pointer pMyData,
        SceFont_t_handle FileID,
        SceFont_t_u32 Offset
);
SceFont_t_error errorCode;
SceFont_t_libId libID;
SceFont_t_initRec initParam = {
        NULL, /* Pointer to user data */
        4, /* Maximum number of fonts that are open simultaneously */
        NULL, /* Handle for cache instance */
        cb_Alloc, /* Memory allocation function */
        cb_Free,  /* Memory release function */
        cb_Open, /* Open*/
        cb_Close, /* Close */
        cb_Read, /* Read */
        cb_Seek, /* Seek */
        NULL, /* Callback when an error occurs */
        NULL, /* Callback when reading ends */
};

/* Generate a libpgf library instance */
libID = sceFontNewLib (&initParam, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontNewLib): 0x%8.8x\n", (int)errorCode);
}
```

Notes

Many libpgf functions receive this *`libID`* in the first argument.

See Also

`SceFont_t_libId`, `SceFont_t_initRec`, `SceFont_t_error`, `sceFontDoneLib()`

sceFontDoneLib

Destroy libpgf library instance

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontDoneLib (
        SceFont_t_libId libID
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Pointer to library instance
errorCode	Error code

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function forcibly closes all fonts that remain open in relation to the one library instance specified by *libID*, releases all memory that had been allocated, and terminates that library instance.

Examples

```
SceFont_t_error errorCode;

errorCode = sceFontDoneLib (libID);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontDoneLib): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontNewLib()`

sceFontSetResolution

Set system resolution

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontSetResolution (
        SceFont_t_libId libID,
        SceFont_t_f32 hResolution,
        SceFont_t_f32 vResolution
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
hResolution	Horizontal resolution value (dpi value)
vResolution	Vertical resolution value (dpi value)

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

libpgf gets the vertical and horizontal resolution information of the display device from the PlayStation®Vita system registry. The `sceFontSetResolution()` function resets the resolution information that libpgf obtained from the PlayStation®Vita system.

Examples

```
SceFont_t_error errorCode;

errorCode = sceFontSetResolution
         (libID, (SceFont_t_f32)256.0f, (SceFont_t_f32)256.0f);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontSetResolution): 0x%8.8x\n", (int)errorCode);
}
```

Notes

This function overwrites only the value that the libpgf instance holds internally. The system registry information that the PlayStation®Vita has is not changed.

See Also

`SceFont_t_libId`, `SceFont_t_error`

sceFontGetNumFontList

Get number of fonts installed on PlayStation®Vita

Definition

```
#include <font/libpgf.h>
SceFont_t_int numFontLists = sceFontGetNumFontList (
        SceFont_t_libId libID,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
errorCode	Address for storing the error code

Return Values

If the function completes normally, the number of fonts that are installed on the PlayStation®Vita is returned.

If an error occurs, 0 is returned in *numFontLists* and one of the following is stored in **errorCode*.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function returns the number of fonts installed on the PlayStation®Vita system.

Examples

```
SceFont_t_error errorCode;
int numFontList;

/* Get number of font lists */
numFontList = sceFontGetNumFontList (libID, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetNumFontList): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontGetFontList()`

sceFontGetFontList

Create list related to fonts installed on PlayStation®Vita

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontGetFontList (
        SceFont_t_libId libID,
        SceFont_t_fontStyleInfo *fontStyleInfo,
        SceFont_t_int arraySize
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
fontStyleInfo	Address of font style information array
arraySize	Size of *fontStyleInfo* array

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function obtains information about fonts installed on the PlayStation®Vita system.

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontStyleInfo aFontStyleInfoList [20];

/* Get font list */
errorCode = sceFontGetFontList
        (libID, aFontStyleInfoList,
        sizeof (aFontStyleInfoList ) / sizeof (aFontStyleInfoList  [0]));
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetFontList): 0x%8.8x\n", (int)errorCode);
}
```

Notes

If the number of fonts that are installed on the PlayStation®Vita system is greater than *`arraySize`*, *`arraySize`* pieces of information from the start of the fonts managed by the PlayStation®Vita system are stored in *`fontStyleInfo`*.

See Also

`SceFont_t_libId`, `SceFont_t_fontStyleInfo`, `SceFont_t_error`, `sceFontGetNumFontList()`

sceFontFindOptimumFont

Find optimum font

Definition

```
#include <font/libpgf.h>
SceFont_t_fontIndex fontIndex = sceFontFindOptimumFont (
        SceFont_t_libId libID,
        SceFont_t_fontStyleInfo *fontStyleInfo,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
fontStyleInfo	Style information of font to be obtained
errorCode	Address for storing error code

Return Values

If the function completes normally, the index value of the optimum font is returned.

If an error occurs, 0 is returned in *fontIndex* and one of the following is stored in **errorCode*.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function finds the nearest font to the font style set in *fontStyleInfo* from the fonts that are installed on the PlayStation®Vita and returns a value for accessing that font in *fontIndex*.

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontStyleInfo targetStyle;
SceFont_t_fontIndex targetFontIndex;

targetStyle.languageCode = SCE_FONT_LANGUAGE_J;
targetStyle.familyCode = SCE_FONT_DEFAULT_FAMILY_CODE;
targetStyle.style = SCE_FONT_DEFAULT_STYLE_CODE;
targetStyle.hSize = targetStyle.vSize = (SceFont_t_f32)10.0f;
targetFontIndex
        = sceFontFindOptimumFont (libID, &targetStyle, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontFindOptimumFont): 0x%8.8x\n", (int)errorCode);
}
```

Notes

This function finds the font determined to be closest based on an internal decision criterion that libpgf has.

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontFindFont()`

sceFontFindFont

Find font

Definition

```
#include <font/libpgf.h>
SceFont_t_fontIndex fontIndex = sceFontFindFont (
        SceFont_t_libId libID,
        SceFont_t_fontStyleInfo *fontStyleInfo,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
fontStyleInfo	Style information about the font to be obtained
errorCode	Address for storing error code

Return Values

If the function completes normally, the index value of the optimum font is returned.

If an error occurs, 0 is returned in *fontIndex* and one of the following is stored in **errorCode*.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function finds the font that conforms to the font style that was set in *fontStyleInfo* from the fonts that are installed on the PlayStation®Vita and returns a value for accessing that font in *fontIndex*.

If no font conforms, -1 is returned in *fontIndex*.

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontStyleInfo targetStyle;
SceFont_t_fontIndex targetFontIndex;

targetStyle.languageCode = SCE_FONT_LANGUAGE_J;
targetStyle.familyCode = SCE_FONT_DEFAULT_FAMILY_CODE;
targetStyle.style = SCE_FONT_DEFAULT_STYLE_CODE;
targetStyle.hSize = targetStyle.vSize = (SceFont_t_f32)7.0f;
targetFontIndex = sceFontFindFont (libID, &targetStyle, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontFindFont): 0x%8.8x\n", (int)errorCode);
}
```

Notes

This function finds the font for which all *`fontStyleInfo`* information matches.

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontFindOptimumFont()`

sceFontGetFontInfoByIndexNumber

Get font information (by specifying font by number)

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontGetFontInfoByIndexNumber (
        SceFont_t_libId libID,
        SceFont_t_fontStyleInfo *fontStyleInfo,
        SceFont_t_fontIndex fontIndex
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
fontStyleInfo	Pointer to area for storing font style information
fontIndex	Font index number

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function gets the font style information of the font that corresponds to the font index number that was returned by `sceFontFindOptimumFont()` or `sceFontFindFont()`.

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontStyleInfo fontStyleInfo;

errorCode = sceFontGetFontInfoByIndexNumber
        (libID, &fontStyleInfo, (SceFont_t_fontIndex)0);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetFontInfoByIndexNumber): 0x%8.8x\n",
               (int)errorCode);
}
```

Notes

When 0 is specified for *`fontIndex`*, the font style information of the PlayStation®Vita system default font is stored in *`fontStyleInfo`*.

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontFindOptimumFont()`, `sceFontFindFont()`

sceFontOpen

Open font (open font that PlayStation®Vita system has internally)

Definition

```
#include <font/libpgf.h>
SceFont_t_fontId fontID = sceFontOpen (
        SceFont_t_libId libID,
        SceFont_t_fontIndex fontIndex,
        SceFont_t_u32 mode,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
fontIndex	Font index number
mode	Access mode One of the following values is assigned. SCE_FONT_FILEBASEDSTREAM, SCE_FONT_MEMORYBASEDSTREAM
errorCode	Address for storing the error code

Return Values

If the function completes normally, an ID for accessing the font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

SCE_FONT_ERROR_NOMEMORY, SCE_FONT_ERROR_LIBID, SCE_FONT_ERROR_ARG, SCE_FONT_ERROR_NOFILE, SCE_FONT_ERROR_FILEOPEN, SCE_FONT_ERROR_FILECLOSE, SCE_FONT_ERROR_FILEREAD, SCE_FONT_ERROR_FILESEEK, SCE_FONT_ERROR_TOOMANYOPENED, SCE_FONT_ERROR_ILLEGALVERSION, SCE_FONT_ERROR_DATAINCONSISTENT, SCE_FONT_ERROR_EXPIRED, SCE_FONT_ERROR_REGISTRY, SCE_FONT_ERROR_NOSUPPORT, SCE_FONT_ERROR_UNKNOWN

Description

This function opens the font corresponding to the font index number that was returned by sceFontFindOptimumFont() or sceFontFindFont().

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontId fontID;

fontID = sceFontOpen
        (libID, targetFontIndex, SCE_FONT_FILEBASEDSTREAM, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontOpen)(%d): 0x%8.8x\n", i, (int)errorCode);
}
```

Notes

If 0 is specified for *`fontIndex`*, the PlayStation®Vita system PSP™-compatible grayscale dot default font is opened.

The file access functions (*`openFunc`*, *`closeFunc`*, *`readFunc`*, *`seekFunc`*) specified by `SceFont_t_initRec` are not used to access a font that is opened by `sceFontOpen()`.

The common `sceFontClose()` function is used to close the font.

See Also

`SceFont_t_initRec`, `SceFont_t_libId`, `SceFont_t_fontId`, `SceFont_t_error`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `sceFontOpenUserFile()`, `sceFontOpenUserMemory()`, `sceFontClose()`

sceFontOpenUserFile

Open font (by specifying filename)

Definition

```
#include <font/libpgf.h>
SceFont_t_fontId fontID = sceFontOpenUserFile (
        SceFont_t_libId libID,
        SceFont_t_pointer filename,
        SceFont_t_u32 mode,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
filename	Font filename
mode	Access mode One of the following values is assigned. `SCE_FONT_FILEBASEDSTREAM`, `SCE_FONT_MEMORYBASEDSTREAM`
errorCode	Address for storing the error code

Return Values

If the function completes normally, an ID for accessing the font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`,
`SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`,
`SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`,
`SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`,
`SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`,
`SCE_FONT_ERROR_UNKNOWN`

Description

This function opens the font file specified by *filename*.

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontId fontID;

fontID = sceFontOpenUserFile
         (libID,
          "host0:FW-NR-DB-JPN.128dpi.7pt.PGF",
          SCE_FONT_FILEBASEDSTREAM, &errorCode);
if ( errorCode != SCE_OK ) {
         printf ("Error (sceFontOpen)(%d): 0x%8.8x\n", i, (int)errorCode);
}
```

Notes

When `SCE_FONT_FILEBASEDSTREAM` is specified for *`mode`*, the file access functions specified by `SceFont_t_initRec` (*`openFunc`*, *`closeFunc`*, *`readFunc`*, *`seekFunc`*) are used to access a font that is opened by `sceFontOpenUserFile()`.

When `SCE_FONT_MEMORYBASEDSTREAM` is specified for *`mode`*, rather than using the file access functions specified by `SceFont_t_initRec` (*`openFunc`*, *`closeFunc`*, *`readFunc`*) to access a font that is opened by `sceFontOpenUserFile()`, `sceIoOpen()`, `sceIoRead()`, and `sceIoClose()` are called directly from libpgf instead.

The common `sceFontClose()` function is used to close the font.

See Also

`SceFont_t_initRec`, `SceFont_t_libId`, `SceFont_t_fontId`, `SceFont_t_error`, `sceFontOpen()`, `sceFontOpenUserMemory()`, `sceFontClose()`

sceFontOpenUserMemory

Open font (by specifying memory address)

Definition

```
#include <font/libpgf.h>
SceFont_t_fontId fontID = sceFontOpenUserMemory (
        SceFont_t_libId libID,
        SceFont_t_pointer addr,
        SceFont_t_u32 size,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
addr	Address of font data that is in memory
size	Size of font data that is in memory
errorCode	Address for storing the error code

Return Values

If the function completes normally, an ID for accessing the font is returned.

If an error occurs, NULL is returned in *fontID* and one of the following is stored in **errorCode*.

SCE_FONT_ERROR_NOMEMORY, SCE_FONT_ERROR_LIBID, SCE_FONT_ERROR_ARG, SCE_FONT_ERROR_NOFILE, SCE_FONT_ERROR_FILEOPEN, SCE_FONT_ERROR_FILECLOSE, SCE_FONT_ERROR_FILEREAD, SCE_FONT_ERROR_FILESEEK, SCE_FONT_ERROR_TOOMANYOPENED, SCE_FONT_ERROR_ILLEGALVERSION, SCE_FONT_ERROR_DATAINCONSISTENT, SCE_FONT_ERROR_EXPIRED, SCE_FONT_ERROR_REGISTRY, SCE_FONT_ERROR_NOSUPPORT, SCE_FONT_ERROR_UNKNOWN

Description

This function opens a font that was read into memory specified by *addr*.

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontId fontID;
SceFont_t_pointer pFontData;
SceFont_t_u32 fontDataSize;

/* For example, function for allocating memory according to file size and reading
file */
fontDataSize = myReadAndAlloc ("host0:FW-NR-DB-JPN.128dpi.7pt.PGF",
&pFontData);

if ( 0 < fontDataSize )
        fontID = sceFontOpenUserMemory
              (libID,
               pFontdata,
               fontDataSize, &errorCode);
        if ( errorCode != SCE_OK ) {
              printf ("Error (sceFontOpen)(%d): 0x%8.8x\n", i,
(int)errorCode);
        }
}
```

Notes

The common `sceFontClose()` function is used to close the font.

See Also

`SceFont_t_libId`, `SceFont_t_fontId`, `SceFont_t_error`, `sceFontOpen()`, `sceFontOpenUserFile()`, `sceFontClose()`

sceFontClose

Close font

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontClose (
        SceFont_t_fontId fontID
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

fontID	Font ID

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function closes a font.

Examples

```
SceFont_t_error errorCode;

errorCode = sceFontClose (fontID);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontClose): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_fontId`, `SceFont_t_error`, `sceFontFindOptimumFont()`, `sceFontFindFont()`, `sceFontOpen()`

sceFontFlush

Clear libpgf internal local cache

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontFlush (
        SceFont_t_fontId fontID
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

fontID	Font ID

Return Values

When the function completes normally, `SCE_OK` is returned.

If an error occurs, `SCE_FONT_ERROR_ARG` is returned.

Description

This function discards libpgf's internal local and short-term cache data and forcibly releases the memory that libpgf acquired for the cache according to the *allocFunc* member variable of `SceFont_t_initRec`.

Examples

```
SceFont_t_error errorCode;

errorCode = sceFontFlush (fontID);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontDoneLib): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_initRec`

sceFontGetFontInfo

Get font information

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontGetFontInfo (
        SceFont_t_fontId fontID,
        SceFont_t_fontInfo *fontInfo
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

fontID	Font ID
fontInfo	Pointer to area for storing font information

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function stores information related to the open font indicated by *fontID* in *fontInfo*.

Examples

```
SceFont_t_error errorCode;
SceFont_t_fontInfo fontInfo;

errorCode = sceFontGetFontInfo (fontID, &fontInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetFontInfo): 0x%8.8x\n", (int)errorCode);
}
```

Notes

The following values related to a font that was opened by `sceFontOpenUserFile()` or `sceFontOpenUserMemory()` cannot be obtained.

fontInfo->*fontStyleInfo*.*weight*
fontInfo->*fontStyleInfo*.*familyCode*
fontInfo->*fontStyleInfo*.*style*
fontInfo->*fontStyleInfo*.*subStyle*
fontInfo->*fontStyleInfo*.*languageCode*
fontInfo->*fontStyleInfo*.*regionCode*
fontInfo->*fontStyleInfo*.*countryCode*
fontInfo->*fontStyleInfo*.*fontName*
fontInfo->*fontStyleInfo*.*fileName*
fontInfo->*fontStyleInfo*.*extraAttributes*
fontInfo->*fontStyleInfo*.*expireDate*

See Also

`SceFont_t_fontId`, `SceFont_t_fontInfo`, `SceFont_t_error`, `sceFontOpen()`

sceFontGetCharInfo

Get character information

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontGetCharInfo (
        SceFont_t_fontId fontID,
        SceFont_t_charCode charCode,
        SceFont_t_charInfo *charInfo
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

fontID	Font ID
charCode	UCS2 character code
charInfo	Character information

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function stores character information related to the character code indicated by *charCode* of the open font indicated by *fontID* in *charInfo*.

Examples

```
SceFont_t_error errorCode;
SceFont_t_charInfo charInfo;
SceFont_t_charCode charCode = 0x90a3;

errorCode = sceFontGetCharInfo (fontID, charCode, &charInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetCharInfo): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_fontId`, `SceFont_t_charCode`, `SceFont_t_charInfo`, `SceFont_t_error`, `sceFontOpen()`

sceFontGetCharImageRect

Get size of character glyph image rectangle

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontGetCharImageRect (
        SceFont_t_fontId fontID,
        SceFont_t_charCode charCode,
        SceFont_t_irect *rect
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

fontID	Font ID
charCode	UCS2 character code
rect	Rectangle information

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function stores the size of the glyph image rectangle of the character code indicated by *charCode* of the open font indicated by *fontID* in *rect*.

Examples

```
SceFont_t_error errorCode;
SceFont_t_charCode charCode = 0x90a3;
SceFont_t_irect rect;

errorCode = sceFontGetCharImageRect (fontID, charCode, &rect);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetCharImageRect): 0x%8.8x\n",
(int)errorCode);
}
```

See Also

`SceFont_t_fontId`, `SceFont_t_charCode`, `SceFont_t_charInfo`, `SceFont_t_error`, `sceFontOpen()`

sceFontGetCharGlyphImage

Get character glyph image

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontGetCharGlyphImage (
        SceFont_t_fontId fontID,
        SceFont_t_charCode charCode,
        SceFont_t_userImageBufferRec *imageBuffer
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

fontID	Font ID
charCode	UCS2 character code
imageBuffer	Pointer to area where information related to buffer for storing glyph image is stored

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function stores the glyph image of the character code indicated by *`charCode`* of the open font indicated by *`fontID`* at the specified position of the buffer indicated by *`imageBuffer`*.

Examples

```
#define IMAGE_WIDTH (256)
#define IMAGE_HEIGHT (128)
SceFont_t_error errorCode;
SceFont_t_charCode charCode = 0x90a3;
SceFont_t_userImageBufferRec imageBufferInfo;

imageBufferInfo.pixelFormat = SCE_FONT_USERIMAGE_DIRECT4_L;
imageBufferInfo.rect.width = IMAGE_WIDTH;
imageBufferInfo.rect.height = IMAGE_HEIGHT;
imageBufferInfo.bytesPerLine = IMAGE_WIDTH / 2;
imageBufferInfo.reserved = 0;
imageBufferInfo.buffer
        = (SceFont_t_u8 *)memalign (16, IMAGE_WIDTH * IMAGE_HEIGHT);
errorCode = sceFontGetCharGlyphImage (fontID, charCode, &imageBufferInfo);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetCharGlyphImage): 0x%8.8x\n",
(int)errorCode);
}
```

See Also

`SceFont_t_fontId`, `SceFont_t_charCode`, `SceFont_t_userImageBufferRec`, `SceFont_t_error`, `sceFontOpen()`

sceFontGetCharGlyphImage_Clip

Get character glyph image with rectangle clipping function

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontGetCharGlyphImage_Clip (
        SceFont_t_fontId fontID,
        SceFont_t_charCode charCode,
        SceFont_t_userImageBufferRec *imageBuffer,
        SceFont_t_s32 clipX,
        SceFont_t_s32 clipY,
        SceFont_t_u32 clipWidth,
        SceFont_t_u32 clipHeight
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

fontID	Font ID
charCode	UCS2 character code
imageBuffer	Pointer to area where information related to buffer for storing glyph image is stored
clipX	X-position of clipping rectangle
clipY	Y-position of clipping rectangle
clipWidth	Clipping rectangle width
clipHeight	Clipping rectangle height

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOFILE`, `SCE_FONT_ERROR_FILEOPEN`, `SCE_FONT_ERROR_FILECLOSE`, `SCE_FONT_ERROR_FILEREAD`, `SCE_FONT_ERROR_FILESEEK`, `SCE_FONT_ERROR_TOOMANYOPENED`, `SCE_FONT_ERROR_ILLEGALVERSION`, `SCE_FONT_ERROR_DATAINCONSISTENT`, `SCE_FONT_ERROR_EXPIRED`, `SCE_FONT_ERROR_REGISTRY`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function stores the glyph image of the character code indicated by *charCode* of the open font indicated by *fontID* at the specified position of the buffer indicated by *imageBuffer* while clipping it according to *clipX*, *clipY*, *clipWidth*, and *clipHeight*.

Examples

```
#define IMAGE_WIDTH (256)
#define IMAGE_HEIGHT (128)
SceFont_t_error errorCode;
SceFont_t_charCode charCode = 0x90a3;
SceFont_t_userImageBufferRec imageBufferInfo;

imageBufferInfo.pixelFormat = SCE_FONT_USERIMAGE_DIRECT4_L;
imageBufferInfo.rect.width = IMAGE_WIDTH;
imageBufferInfo.rect.height = IMAGE_HEIGHT;
imageBufferInfo.bytesPerLine = IMAGE_WIDTH / 2;
imageBufferInfo.reserved = 0;
imageBufferInfo.buffer
        = (SceFont_t_u8 *)memalign (16, IMAGE_WIDTH * IMAGE_HEIGHT);
errorCode = sceFontGetCharGlyphImage_Clip
         (fontID, charCode, &imageBufferInfo, 3, 3, 10, 10);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontGetCharGlyphImage_Clip): 0x%8.8x\n",
(int)errorCode);
}
```

See Also

SceFont_t_fontId, SceFont_t_charCode, SceFont_t_userImageBufferRec, SceFont_t_error, sceFontOpen()

sceFontPixelToPointH

Convert from pixels to points (values related to horizontal direction)

Definition

```
#include <font/libpgf.h>
SceFont_t_f32 point = sceFontPixelToPointH (
        SceFont_t_libId libID,
        SceFont_t_f32 pixel,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
pixel	Value in pixel units
errorCode	Address for storing the error code

Return Values

If the function completes normally, the result when *pixel* is unit converted to points is returned.

If an error occurs, 0 is returned in *point* and one of the following is stored in **errorCode*.

SCE_FONT_ERROR_NOMEMORY, SCE_FONT_ERROR_LIBID, SCE_FONT_ERROR_ARG, SCE_FONT_ERROR_NOSUPPORT, SCE_FONT_ERROR_UNKNOWN

Description

This function converts the value indicated by *pixel*, which is in dot units, to the point value according to the horizontal resolution (dpi value) for PSP™-compatible grayscale dot that the PlayStation®Vita system has.

Examples

```
SceFont_t_error errorCode;
SceFont_t_f32 point;

point = sceFontPixelToPointH (libID, (SceFont_t_f32)18.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontPixelToPointH): 0x%8.8x\n", (int)errorCode);
}
```

See Also

SceFont_t_libId, SceFont_t_error, sceFontPixelToPointV(), sceFontPointToPixelH(), sceFontPointToPixelV(), sceFontSetResolution(), SceFont_t_fontStyleInfo

sceFontPixelToPointV

Convert from pixels to points (values related to vertical direction)

Definition

```
#include <font/libpgf.h>
SceFont_t_f32 point = sceFontPixelToPointV (
        SceFont_t_libId libID,
        SceFont_t_f32 pixel,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
pixel	Value in pixel units
errorCode	Address for storing the error code

Return Values

If the function completes normally, the result when `pixel` is unit converted to points is returned.

If an error occurs, 0 is returned in `point` and one of the following is stored in `*errorCode`.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function converts the value indicated by `pixel`, which is in dot units, to the point value according to the vertical resolution (dpi value) for PSP™-compatible grayscale dot that the PlayStation®Vita system has.

Examples

```
SceFont_t_error errorCode;
SceFont_t_f32 point;

point = sceFontPixelToPointV (libID, (SceFont_t_f32)18.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontPixelToPointV): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontPixelToPointH()`, `sceFontPointToPixelH()`, `sceFontPointToPixelV()`, `sceFontSetResolution()`, `SceFont_t_fontStyleInfo`

sceFontPointToPixelH

Convert from points to pixels (values related to horizontal direction)

Definition

```
#include <font/libpgf.h>
SceFont_t_f32 pixel = sceFontPointToPixelH (
        SceFont_t_libId libID,
        SceFont_t_f32 point,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
point	Value in point units
errorCode	Address for storing the error code

Return Values

If the function completes normally, the result when *point* is unit converted to pixels is returned.

If an error occurs, 0 is returned in *pixel* and one of the following is stored in **errorCode*.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function converts the value indicated by *point*, which is in point units, to the pixel value according to the horizontal resolution (dpi value) for PSP™-compatible grayscale dot that the PlayStation®Vita system has.

Examples

```
SceFont_t_error errorCode;
SceFont_t_f32 pixel;

pixel = sceFontPointToPixelH (libID, (SceFont_t_f32)10.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontPointToPixelH): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontPixelToPointH()`, `sceFontPixelToPointV()`, `sceFontPointToPixelV()`, `sceFontSetResolution()`, `SceFont_t_fontStyleInfo`

sceFontPointToPixelV

Convert from points to pixels (values related to vertical direction)

Definition

```
#include <font/libpgf.h>
SceFont_t_f32 pixel = sceFontPointToPixelV (
        SceFont_t_libId libID,
        SceFont_t_f32 point,
        SceFont_t_error *errorCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
point	Value in point units
errorCode	Address for storing the error code

Return Values

If the function completes normally, the result when `point` is unit converted to pixels is returned.

If an error occurs, 0 is returned in `pixel` and one of the following is stored in `*errorCode`.

`SCE_FONT_ERROR_NOMEMORY`, `SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function converts the value indicated by `point`, which is in point units, to the pixel value according to the vertical resolution (dpi value) for PSP™-compatible grayscale dot that the PlayStation®Vita system has.

Examples

```
SceFont_t_error errorCode;
SceFont_t_f32 pixel;

pixel = sceFontPointToPixelV (libID, (SceFont_t_f32)10.0f, &errorCode);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontPointToPixelV): 0x%8.8x\n", (int)errorCode);
}
```

See Also

`SceFont_t_libId`, `SceFont_t_error`, `sceFontPixelToPointH()`, `sceFontPixelToPointV()`, `sceFontPointToPixelH()`, `sceFontSetResolution()`, `SceFont_t_fontStyleInfo`

sceFontSetAltCharacterCode

Set alternate character code

Definition

```
#include <font/libpgf.h>
SceFont_t_error errorCode = sceFontSetAltCharacterCode (
        SceFont_t_libId libID,
        SceFont_t_charCode charCode
);
```

Calling Conditions

Cannot be called from an interrupt handler

Can be called from a thread (must be called in an interrupt-enabled state)

Not multithread safe

Arguments

libID	Library instance pointer
charCode	Character code of alternate characters

Return Values

If the function completes normally, `SCE_OK` is returned.

If an error occurs, one of the following is returned.

`SCE_FONT_ERROR_LIBID`, `SCE_FONT_ERROR_ARG`, `SCE_FONT_ERROR_NOSUPPORT`, `SCE_FONT_ERROR_UNKNOWN`

Description

This function specifies the character code of the characters to be used as alternate glyphs when an attempt is made to get the glyph image by using a function such as `sceFontGetCharGlyphImage()` for a character code for which no glyphs exist.

Immediately after an instance of the libpgf library is created, U+005F is set as the default alternate character code.

When a function such as `sceFontGetCharGlyphImage()` is used to access a font and the font to be accessed does not contain the character with the character code that was passed to the function, libpgf automatically replaces the character with the character of the character code that was set by `sceFontSetAltCharacterCode()`.

However, a precondition for this to work is that the character set of the font that is to be accessed by a function such as `sceFontGetCharGlyphImage()` contains the character of the character code *charCode* that was specified by `sceFontSetAltCharacterCode()`.

If the font does not contain that alternate character code, replacement by an alternate character is not performed, and the result is handled as a character that has both its width and height set to zero.

The default alternate character code U+005F is included in both the built-in Latin fonts and built-in Japanese fonts, but it is not included in the built-in internal Korean fonts. This means that when a built-in Korean font is used with the default alternate character, alternate character replacement is not performed.

For details about built-in font character sets, refer to the "PlayStation®Vita FONT CHARACTER TABLE".

Examples

```
SceFont_t_error errorCode;
SceFont_t_f32 pixel;

errorCode = sceFontSetAltCharacterCode (libID, (SceFont_t_charCode)0x22a0);
if ( errorCode != SCE_OK ) {
        printf ("Error (sceFontSetAltCharacterCode): 0x%8.8x\n",
(int)errorCode);
}
```

See Also

SceFont_t_libId, SceFont_t_error,
SceFont_t_charCode,sceFontGetCharGlyphImage()

Constants

List of Error Codes

libpgf Error Codes

Definition

Macro	Value	Description
`SCE_OK`	0	No error (normal termination)
`SCE_FONT_ERROR_NOMEMORY`	0x80460001	Memory allocation failed
`SCE_FONT_ERROR_LIBID`	0x80460002	Invalid library instance
`SCE_FONT_ERROR_ARG`	0x80460003	Invalid argument
`SCE_FONT_ERROR_NOFILE`	0x80460004	No file
`SCE_FONT_ERROR_FILEOPEN`	0x80460005	Processing for opening file failed
`SCE_FONT_ERROR_FILECLOSE`	0x80460006	Processing for closing file failed
`SCE_FONT_ERROR_FILEREAD`	0x80460007	Processing for reading file failed
`SCE_FONT_ERROR_FILESEEK`	0x80460008	File seeking failed
`SCE_FONT_ERROR_TOOMANYOPENED`	0x80460009	Too many open fonts
`SCE_FONT_ERROR_ILLEGALVERSION`	0x8046000a	Unsupported font version
`SCE_FONT_ERROR_DATAINCONSISTENT`	0x8046000b	Inconsistency in font data
`SCE_FONT_ERROR_EXPIRED`	0x8046000c	Usage period expired
`SCE_FONT_ERROR_REGISTRY`	0x8046000d	Cause related to system registry
`SCE_FONT_ERROR_NOSUPPORT`	0x8046000e	Unsupported cause
`SCE_FONT_ERROR_UNKNOWN`	0x8046ffff	Unknown error