Display Service Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

Constant Definitions

SCE_DISPLAY_PIXELFORMAT_A8B8G8R8

RGBA8888 32-bit color output mode constant

Definition

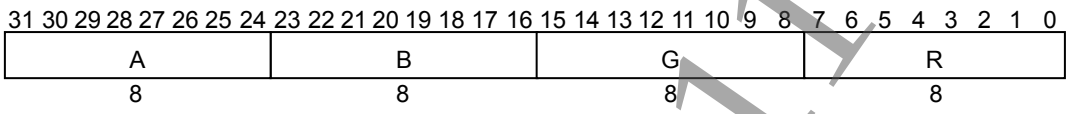
```
#include <display.h>
#define SCE_DISPLAY_PIXELFORMAT_A8B8G8R8          0x0000000U
```

Description

This constant represents output in the R:G:B:A=8:8:8:8 pixel format.

Specify it using `sceDisplaySetFrameBuf()` when performing display output of graphics drawn in the R:G:B:A=8:8:8:8 pixel format.

R, G, B, A assignments in the pixel data are as follows.



See Also

`sceDisplaySetFrameBuf()`

Datatypes

SceDisplayFrameBuf

Frame Buffer Information

Definition

```
#include <display.h>
typedef struct SceDisplayFrameBuf {
        SceUInt32 size;
        void *base;
        SceUInt32 pitch;
        SceUInt32 pixelformat;
        SceUInt32 width;
        SceUInt32 height;
} SceDisplayFrameBuf;
```

Members

size	`SceDisplayFrameBuf` structure size. Set `sizeof(SceDisplayFrameBuf)`
base	Base address of the frame buffer to be displayed Allocate the frame buffer on the CDRAM and set with 256 byte alignment
pitch	Horizontal pixel count of the frame buffer (line size) Set to an integral multiple of 64 pixels.
pixelformat	Pixel format Specify `SCE_DISPLAY_PIXELFORMAT_A8B8G8R8`
width	Horizontal pixel count of the frame buffer (display area size)
height	Vertical pixel count of the frame buffer (display area size)

Description

This structure describes the components of the frame buffer set with the `sceDisplaySetFrameBuf()` function.

Set structure size `sizeof(SceDisplayFrameBuf)` in the *size* member.

The resolution size of the frame buffer specified in *width*, *height* can be one of the following: 480 x 272, 640 x 368, 720 x 408, or 960 x 544.

Since the display resolution of the screen (touchscreen) is 960 x 544, if a resolution other than 960 x 544 is specified, display may be scaled by the hardware.

See Also

`sceDisplaySetFrameBuf(),sceDisplayGetFrameBuf()`

Mode Setting Functions

sceDisplayGetRefreshRate

Get number of frames per second

Definition

```
#include <display.h>
SceInt32 sceDisplayGetRefreshRate (
        float *pFps
);
```

Calling Conditions

Multithread safe.

Arguments

pFps Pointer of type `float *` to obtain the number of frames per second

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
<0	Error

Description

This function obtains the theoretical number of frames per second in the current screen mode.

The number of frames per second on the screen (touchscreen) is NTSC-compatible 59.94005994... Hz.

Note that `sceDisplayGetRefreshRate()` function returns a theoretical value, and that some error may be present relative to the actual time.

sceDisplaySetFrameBuf

Update frame buffer base address

Definition

```
#include <display.h>
SceInt32 sceDisplaySetFrameBuf (
        const SceDisplayFrameBuf *pFrameBuf,
        SceInt32 iUpdateTimingMode
);
```

Calling Conditions

Multithread safe.

Arguments

pFrameBuf	Start address of frame buffer (if NULL, output black)
iUpdateTimingMode	Specification of frame buffer start address update timing

Values which Can Be Specified for *iUpdateTimingMode*

Value	Description
SCE_DISPLAY_UPDATETIMING_NEXTHSYNC	Update upon next HSYNC
SCE_DISPLAY_UPDATETIMING_NEXTVSYNC	Update during next VBLANK interval

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
SCE_DISPLAY_ERROR_INVALID_VALUE	The *size* member indicating the structure of the frame buffer specified in the argument *pFrameBuf* is invalid
SCE_DISPLAY_ERROR_INVALID_ADDR	The *base* member indicating the base address of the frame buffer specified in the argument *pFrameBuf* is invalid
SCE_DISPLAY_ERROR_INVALID_PIXELFORMAT	The *pixelformat* member indicating the pixel format of the frame buffer specified in the argument *pFrameBuf* is invalid
SCE_DISPLAY_ERROR_INVALID_PITCH	The *pitch* member indicating the pixel format of the frame buffer specified in the argument *pFrameBuf* is invalid
SCE_DISPLAY_ERROR_INVALID_RESOLUTION	The *pitch*, *width* and *height* member of the frame buffer specified in the argument *pFrameBuf* is invalid
SCE_DISPLAY_ERROR_INVALID_UPDATETIMING	The frame buffer update timing specified in the argument *iUpdateTimingMode* is invalid
<0	Other error

Description

This function sets the base address (upper-left coordinate), width, and pixel format to be displayed on the screen (touchscreen). Allocate the base address on the CDRAM and set it to 256 byte alignment and set horizontal pixel count to an integral multiple of 64.

When either the width or pixel format of the frame buffer is changed, the update cannot be performed immediately. Make sure to specify that switching be performed during the next VBLANK interval by setting the `SCE_DISPLAY_UPDATETIMING_NEXTVSYNC` to *`iUpdateTimingMode`* argument .

When NULL is set for the *`pFrameBuf`* argument, the output is blacked out.

When switching to or returning from blacked out output, switching must be performed during the next VBLANK interval. Since output is blacked out immediately after process start-up, `SCE_DISPLAY_UPDATETIMING_NEXTVSYNC` is used when displaying first starts.

See Also

`sceDisplayGetFrameBuf()`

sceDisplayGetFrameBuf

Get frame buffer base address

Definition

```
#include <display.h>
SceInt32 sceDisplayGetFrameBuf (
        const SceDisplayFrameBuf *ppFrameBuf,
        SceInt32 iUpdateTimingMode
);
```

Calling Conditions

Multithread safe.

Arguments

ppFrameBuf	Pointer to variable of type `SceDisplayFrameBuf *` for receiving frame buffer address
iUpdateTimingMode	Specifies the address to be obtained by *ppFrameBuf*

Values which Can Be Specified for *iUpdateTimingMode*

Value	Description
SCE_DISPLAY_UPDATETIMING_NEXTHSYNC	Address of frame buffer currently being set
SCE_DISPLAY_UPDATETIMING_NEXTVSYNC	Address of frame buffer currently being set

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
SCE_DISPLAY_ERROR_INVALID_VALUE	The *size* member indicating the structure of the frame buffer specified in the argument *pFrameBuf* is invalid
SCE_DISPLAY_ERROR_INVALID_UPDATETIMING	Invalid frame buffer update timing specified in the argument *iUpdateTimingMode*
<0	Other error

Description

This function obtains the base address (upper left coordinate), width, and pixel format of the frame buffer being displayed on the screen (touchscreen).

In the current implementation, regardless of the value specified to the *iUpdateTimingMode* argument, the frame buffer set with the last `sceDisplaySetFrameBuf()` call will be obtained.

See Also

`sceDisplaySetFrameBuf()`

VBLANK Functions

sceDisplayGetVcount

Get number of VSYNCs

Definition

```
#include <display.h>
SceInt32 sceDisplayGetVcount (
        void
);
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

Value	Description
0 - 0xFFFF	Number of V counts

Description

This function returns the value in which each VSYNC during free running is counted. It increases by 1 at every frame. A wrap-around occurs at 16 bits.

sceDisplayWaitVblankStart, sceDisplayWaitVblankStartCB

Thread wait for start of VBLANK interval

Definition

```
#include <display.h>
SceInt32 sceDisplayWaitVblankStart (
        void
);
SceInt32 sceDisplayWaitVblankStartCB (
        void
);
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
<0	Error

Description

This function places the thread in WAIT state until the start of the next VBLANK interval.

Regardless of whether a VBLANK interval was in progress when the `sceDisplayWaitVblankStart()` or `sceDisplayWaitVblankStartCB()` function was called, the thread enters WAIT state until the start of the next VBLANK interval.

The `sceDisplayWaitVblankStart()` function does not execute a thread manager callback during a WAIT state. If you also want to perform callback processing during the WAIT state, use the `sceDisplayWaitVblankStartCB()` function.

See Also

`sceDisplayWaitVblankStartMulti()`,`sceDisplayWaitVblankStartMultiCB()`

sceDisplayWaitVblankStartMulti, sceDisplayWaitVblankStartMultiCB

Wait for start of multiple VBLANK intervals for each thread

Definition

```
#include <display.h>
SceInt32 sceDisplayWaitVblankStartMulti (
        SceUInt32 uiVcount
);
SceInt32 sceDisplayWaitVblankStartMultiCB (
        SceUInt32 uiVcount
);
```

Calling Conditions

Multithread safe.

Arguments

uiVcount Number of VSYNC cycles to wait for (1 to 65535)

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
<0	Error

Description

This function places the thread in WAIT state until the start of the next VBLANK, after the specified number of VSYNCs has elapsed since the last time a VBLANK wait function was called.

The following VBLANK wait functions keep track of the number of VSYNCs that have occurred for each thread after the respective thread has returned from a wait state.

`sceDisplayWaitVblankStart()`, `sceDisplayWaitVblankStartCB()`, `sceDisplayWaitVblankStartMulti()`, `sceDisplayWaitVblankStartMultiCB()`

The `sceDisplayWaitVblankStartMulti()` and `sceDisplayWaitVblankStartMultiCB()` functions make the respective thread wait until the next VBLANK after the specified number of VSYNCs have elapsed as determined from the VSYNC count that was returned from the last VBLANK wait state, that was recorded for each thread.

For example, if the `sceDisplayWaitVblankStartMulti()` function is called with 2 specified for *uiVcount*, and no VSYNCs have elapsed since the last call, the thread will wait until the start of the second VBLANK. If one VSYNC has elapsed since the last call, the thread will wait until the start of the next VBLANK. If two or more VSYNCs have elapsed since the last call, the thread will start at the next VBLANK. This enables the system to run at a fixed FPS as long as no processing drop occurs. If the function is called with 1 specified for *uiVcount*, the same operations as `sceDisplayWaitVblankStart()` and `sceDisplayWaitVblankStartCB()` will be performed.

The `sceDisplayWaitVblankStartMulti()` function does not execute a thread manager callback during a WAIT state. If you also want to perform callback processing during the WAIT state, use the `sceDisplayWaitVblankStartMultiCB()` function.

Use of the `sceDisplayWaitVblankStartMulti()` or `sceDisplayWaitVblankStartMultiCB()` function is not recommended while a callback function is executing.

See Also

`sceDisplayWaitVblankStart()`,`sceDisplayWaitVblankStartCB()`

sceDisplayWaitSetFrameBuf, sceDisplayWaitSetFrameBufCB

Wait for start of VBLANK interval from the last update of frame buffer

Definition

```
#include <display.h>
SceInt32 sceDisplayWaitSetFrameBuf (
        void
);
SceInt32 sceDisplayWaitSetFrameBufCB (
        void
);
```

Calling Conditions

Multithread safe.

Arguments

None

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
<0	Error

Description

This function places the thread in WAIT state until the start of the next VBLANK interval taking the last update of the display frame buffer performed with `sceDisplaySetFrameBuf()` function as the starting point.

Regardless of whether a VBLANK interval was in progress when the `sceDisplayWaitSetFrameBuf()` or `sceDisplayWaitSetFrameBufCB()` function was called, the thread enters WAIT state until the start of the next VBLANK interval taking the thread on which the display frame buffer was updated with the `sceDisplaySetFrameBuf()` function as the starting point.

In the state in which the frame buffer update has been registered with the `sceDisplaySetFrameBuf()` function, if `sceDisplayWaitSetFrameBuf()` or `sceDisplayWaitSetFrameBufCB()` is called after the start timing of the next VBLANK has elapsed, the thread will not enter WAIT state and will return because the frame buffer has been already updated. In this case, the phase in which the thread returns from this function is undefined in the VSYNC cycles.

The `sceDisplayWaitSetFrameBuf()` function does not execute a thread manager callback during a WAIT state. If you also want to perform callback processing during the WAIT state, use the `sceDisplayWaitSetFrameBufCB()` function.

See Also

`sceDisplayWaitSetFrameBufMulti()`, `sceDisplayWaitSetFrameBufMultiCB()`

sceDisplayWaitSetFrameBufMulti, sceDisplayWaitSetFrameBufMultiCB

Wait for start of multiple VBLANK intervals from the last update of frame buffer

Definition

```
#include <display.h>
SceInt32 sceDisplayWaitSetFrameBufMulti (
        SceUInt32 uiVcount
);
SceInt32 sceDisplayWaitSetFrameBufMultiCB (
        SceUInt32 uiVcount
);
```

Calling Conditions

Multithread safe.

Arguments

uiVcount Number of VSYNC cycles to wait for (1 to 65535)

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
<0	Error

Description

This function places the thread in WAIT state until the start of the next VBLANK, after the specified number of VSYNCs has elapsed, taking the last update of the display frame buffer performed with the `sceDisplaySetFrameBuf()` function as the starting point.

The `sceDisplayWaitSetFrameBufMulti()` and `sceDisplayWaitSetFrameBufMultiCB()` functions make the thread wait until the next VBLANK after the specified number of VSYNCs have elapsed as determined from the VSYNC count that the last frame buffer update was performed.

For example, if the `sceDisplayWaitSetFrameBufMulti()` function is called with 2 specified for *uiVcount*, and no VSYNCs have elapsed since the last update of the frame buffer, the thread will wait until the start of the second VBLANK. If one VSYNC has elapsed since the last update of the frame buffer, the thread will wait until the start of the next VBLANK. If two or more VSYNCs have elapsed since the last update of the frame buffer, the thread will not perform any wait operations. If the function is called with 1 specified for *uiVcount*, the same operations as `sceDisplayWaitSetFrameBuf()` and `sceDisplayWaitSetFrameBufCB()` will be performed.

The `sceDisplayWaitSetFrameBufMulti()` function does not execute a thread manager callback during a WAIT state. If you also want to perform callback processing during the WAIT state, use the `sceDisplayWaitSetFrameBufMultiCB()` function.

See Also

`sceDisplayWaitSetFrameBuf()`, `sceDisplayWaitSetFrameBufCB()`

sceDisplayRegisterVblankStartCallback

Register VBLANK callback function

Definition

```
#include <display.h>
SceInt32 sceDisplayRegisterVblankStartCallback (
        SceUID uid
);
```

Calling Conditions

Multithread safe.

Arguments

uid Callback SceUID

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
<0	Error

Description

This function sets the callback function to be called at the each start of VBLANK.

By registering with the `sceDisplayRegisterVblankStartCallback()` function the SceUID of the callback created with the `sceKernelCreateCallback()` function, the callback is notified at each VBLANK start. The callback function is actually called when the thread that has created the callback is placed in WAIT state by a wait function with "CB" at the end of the function name.

See Also

`vblankcallback()`

sceDisplayUnregisterVblankStartCallback

Unregister VBLANK callback

Definition

```
#include <display.h>
SceInt32 sceDisplayUnregisterVblankStartCallback (
        SceUID uid
);
```

Calling Conditions

Multithread safe.

Arguments

uid Callback SceUID

Return Values

If an error occurs, a negative value is returned.

Value	Description
0	Success
<0	Error

Description

This function unregisters the callback notified at each start of VBLANK.

Please unregister the callback function first if you wish to delete the callback function with `sceKernelDeleteCallback()`.

See Also

`sceDisplayRegisterVblankStartCallback()`

vblankcallback

VBLANK callback function prototype

Definition

```
SceInt32 vblankcallback (
        SceUID notifyId,
        SceInt32 notifyCount,
        SceInt32 notifyArg,
        void *pCommon
);
```

Calling Conditions

Called from callback context

Arguments

notifyId	Callback SceUID value
notifyCount	The number of notification of the callback after the callback function is last called.
notifyArg	Notified 0
pCommon	Cookie value that was set when callback was registered

Return Values

Return 0

Description

This callback function prototype receives the callback notified at each start of VBLANK. By registering with the `sceDisplayRegisterVblankStartCallback()` function the SceUID of the callback created with the `sceKernelCreateCallback()` function, the callback is notified at each VBLANK start. The callback function is actually called when the thread that has created the callback is placed in WAIT state by a wait function with "CB" at the end of the function name.

See Also

`sceKernelCreateCallback(),sceDisplayRegisterVblankStartCallback()`