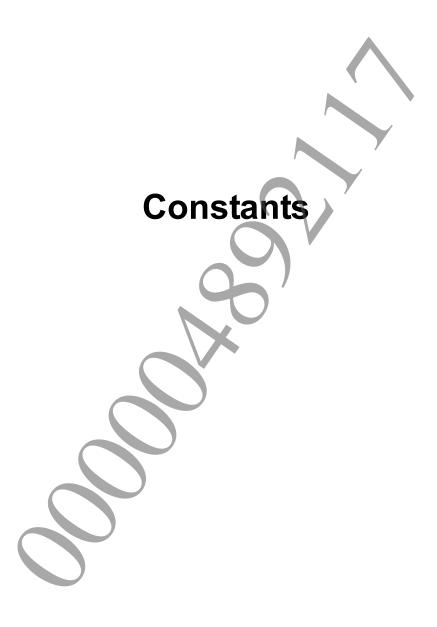


© 2014 Sony Computer Entertainment Inc. All Rights Reserved. SCE Confidential

# **Table of Contents**

Constants	3
Basic Button State Constants	4
SCE_CTRL_INTERCEPTED	6
Controller Mode Constants	7
Wireless Controller Connection State Constants	8
Error Codes	S
Datatypes	10
SceCtrlData	11
SceCtrlRapidFireRule	12
Datatypes for the Wireless Controller	13
SceCtrlData2	
SceCtrlWirelessControllerInfo	
Mode Setting Functions	
sceCtrlSetSamplingMode	
sceCtrlGetSamplingMode	
Data Obtaining Functions	
sceCtrlPeekBufferPositive, sceCtrlPeekBufferNegative	
sceCtrlReadBufferPositive, sceCtrlReadBufferNegative	
Data Obtaining Functions for the Wireless Controller	25
sceCtrlPeekBufferPositive2, sceCtrlPeekBufferNegative2	
sceCtrlReadBufferPositive2, sceCtrlReadBufferNegative2	
Debug Support Functions	30
sceCtrlClearRapidFire	31
sceCtrlSetRapidFire	32
Information Obtaining Functions for the Wireless Controller	
sceCtrlIsMultiControllerSupported	
sceCtrlGetWirelessControllerInfo	



## **Basic Button State Constants**

#### Basic button state constants

#### **Definition**

#include <ctrl.h>
#define Value (Number)

Value	(Number)	Description
SCE_CTRL_SELECT	(1<<0)	SELECT button/SHARE button
SCE_CTRL_L3	(1<<1)	L3 button
SCE_CTRL_R3	(1<<2)	R3 button
SCE_CTRL_START	(1<<3)	START button/OPTIONS button
SCE_CTRL_UP	(1<<4)	Up button
SCE_CTRL_Lup	,	
SCE_CTRL_RIGHT	(1<<5)	Right button
SCE_CTRL_Lright		
SCE_CTRL_DOWN	(1<<6)	Down button
SCE_CTRL_Ldown		
SCE_CTRL_LEFT	(1<<7)	Left button
SCE_CTRL_Lleft		
SCE_CTRL_L	$(1 \le 8)$	L button
SCE_CTRL_R	(1<<9)	R button
SCE_CTRL_L2	(1<<8)	L2 button
SCE_CTRL_R2	(1<<9)	R2 button
SCE_CTRL_L1	(1<<10)	L1 button
SCE_CTRL_R1	(1<<11)	R1 button
SCE_CTRL_TRIANGLE	(1<<12)	Triangle button
SCE_CTRL_Rup		
SCE_CTRL_CIRCLE	(1<<13)	Circle button
SCE_CTRL_Rright		
SCE_CTRL_CROSS	(1<<14)	Cross button
SCE_CTRL_Rdown		
SCE_CTRL_SQUARE	(1<<15)	Square button
SCE_CTRL_Rleft		_

#### **Description**

These constants indicate the bit positions representing the pressed/not pressed state of each basic button to be stored in the <code>buttons</code> member of the <code>SceCtrlData/SceCtrlData2</code> structure, which stores controller state information.

When using sceCtrlPeekBufferPositive(), sceCtrlPeekBufferNegative(), sceCtrlReadBufferPositive() or sceCtrlReadBufferNegative() to obtain SceCtrlData, SCE\_CTRL\_L and SCE\_CTRL\_R will be contained in the buttons member. Note that SCE\_CTRL\_L1, SCE\_CTRL\_R1, SCE\_CTRL\_L2, SCE\_CTRL\_R2, SCE\_CTRL\_L3, and SCE\_CTRL\_R3 will not be contained in the buttons member.

When using sceCtrlPeekBufferPositive2(), sceCtrlPeekBufferNegative2(), sceCtrlReadBufferPositive2() or sceCtrlReadBufferNegative2() to obtain SceCtrlData2, SCE\_CTRL\_L1, SCE\_CTRL\_R1, SCE\_CTRL\_L2, SCE\_CTRL\_R2, SCE\_CTRL\_L3, and SCE\_CTRL\_R3 will be contained in the buttons member. Note that SCE\_CTRL\_L and SCE\_CTRL\_R will not be contained in the buttons member.

#### See Also

SceCtrlData, SceCtrlData2



# SCE CTRL\_INTERCEPTED

#### Controller intercepted constant

#### **Definition**

#include <ctrl.h>
#define SCE CTRL INTERCEPTED (1<<16)</pre>

#### **Description**

This constant indicates the bit position representing controller input interception to be stored in the <code>buttons</code> member of the <code>SceCtrlData/SceCtrlData2</code> structure, which stores controller state information.

#### See Also

SceCtrlData, SceCtrlData2



## **Controller Mode Constants**

#### Controller mode constants

#### **Definition**

```
#include <ctrl.h>
#define SCE_CTRL_MODE_DIGITALONLY 0
#define SCE_CTRL_MODE_DIGITALANALOG 1
#define SCE_CTRL_MODE_DIGITALANALOG WIDE 2
```

#### **Description**

These constants are used when setting the controller mode with scectrlSetSamplingMode().

SCE\_CTRL\_MODE\_DIGITALONLY is a constant that indicates the buttons only mode in which only buttons are used and the analog sticks are not.

The SCE\_CTRL\_MODE\_DIGITALANALOG is a constant that represents the buttons/analog sticks mode with analog sticks operating in the normal mode.

SCE\_CTRL\_MODE\_DIGITALANALOG\_WIDE is a constant that indicates the buttons/analog sticks mode with analog sticks operating in the wide mode.

For details on the operation modes of analog sticks, refer to the "Operation Modes of Analog Sticks" section in the "Controller Service Overview" document.

#### See Also

sceCtrlSetSamplingMode(), sceCtrlGetSamplingMode()



# **Wireless Controller Connection State Constants**

Wireless controller connection state constants

#### **Definition**

#### **Description**

These constants are used when obtaining information of the wireless controller with sceCtrlGetWirelessControllerInfo().

SCE\_CTRL\_WIRELESS\_INFO\_NOT\_CONNECTED represents that a wireless controller is not connected. SCE\_CTRL\_WIRELESS\_INFO\_CONNECTED represents that a wireless controller is connected.

#### See Also

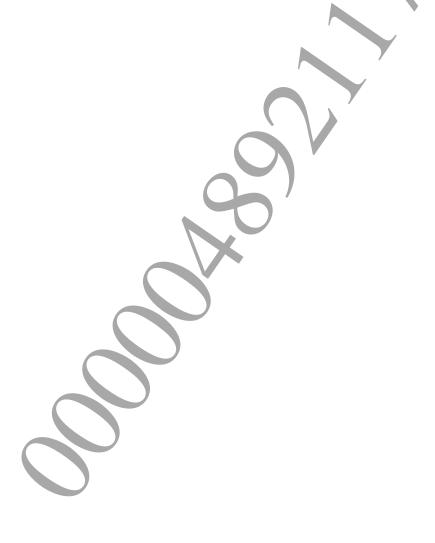
sceCtrlGetWirelessControllerInfo()

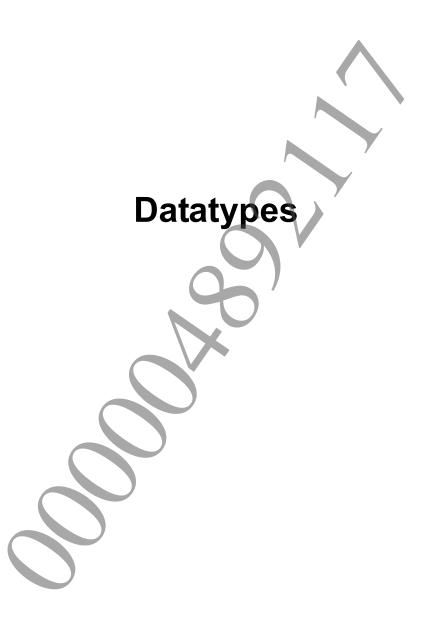
# **Error Codes**

#### Error codes

#### Definition

Value	(Number)	Description
SCE_CTRL_ERROR_INVALID_ARG	0x80340001	Invalid argument specified
SCE_CTRL_ERROR_PRIV_REQUIRED	0x80340002	Access with invalid privilege
SCE_CTRL_ERROR_NO_DEVICE	0x80340020	Specified device does not exist
SCE_CTRL_ERROR_NOT_SUPPORTED	0x80340021	Not supported
SCE_CTRL_ERROR_INVALID_MODE	0x80340022	Invalid mode
SCE CTRL ERROR FATAL	0x803400FF	Fatal error occurred





## **SceCtrlData**

#### Controller state information

#### **Definition**

#### **Members**

timeStamp	Time stamp in the Controller Service (process time: μ sec)
buttons	Button state information
	*Includes SCE_CTRL_L and SCE_CTRL_R
	(excludes SCE_CTRL_L1, SCE_CTRL_R1, SCE_CTRL_L2, SCE_CTRL_R2,
	SCE_CTRL_L3, and SCE_CTRL_R3)
1x	Left stick X axis (left 0x00 to right 0xff)
<i>1y</i>	Left stick Y axis (top 0x00 to bottom 0xff)
rx	Right stick X axis (left 0x00 to right 0xff)
ry	Right stick Y axis (top 0x00 to bottom 0xff)
rsrv	Reserved area

#### **Description**

This structure is for obtaining controller state information.

The time stamp of the time (process time) at which sampling was performed and data was obtained is placed in the timeStamp member.

32-bit button state information is placed in the <code>buttons</code> member. Using the basic button state constants and <code>SCE\_CTRL\_INTERCEPTED</code>, an application can determine the pressed/not pressed state of each button, etc.

The analog stick coordinate data, with 0x80 taken to be the center, is placed in the 1x, 1y, rx and ry members.

#### See Also

```
Basic Button State Constants, SCE_CTRL_INTERCEPTED, sceCtrlPeekBufferPositive(), sceCtrlPeekBufferNegative(), sceCtrlReadBufferPositive(), sceCtrlReadBufferNegative()
```

# SceCtrlRapidFireRule

#### Set button rapid-fire rules

#### **Definition**

#### **Members**

uiMask	Comparison mask of the button operation for rapid-fire trigger
uiTrigger	Button operation for rapid-fire trigger
uiTarget	Button for which rapid-fire input is performed
uiDelay	Dead time of rapid-fire trigger (sampling count)
uiMake	Button press time (sampling count)
uiBreak	Button release time (sampling count)

#### **Description**

This structure is used as the argument for setting the rules for the button rapid-fire functionality with sceCtrlSetRapidFire() mainly for debugging.

For details, refer to sceCtrlSetRapidFire().

#### See Also

sceCtrlSetRapidFire()



## SceCtrlData2

#### Controller state information

#### **Definition**

```
#include <ctrl.h>
typedef struct SceCtrlData2 {
        SceUInt64 timeStamp;
        SceUInt32 buttons;
        SceUInt8 1x;
        SceUInt8 ly;
        SceUInt8 rx;
        SceUInt8 ry;
        SceUInt8 up;
        SceUInt8 right;
        SceUInt8 down;
        SceUInt8 left;
        SceUInt8 12;
        SceUInt8 r2;
        SceUInt8 11;
        SceUInt8 r1;
        SceUInt8 triangle;
        SceUInt8 circle;
        SceUInt8 cross;
        SceUInt8 square;
        SceUInt8 rsrv[4];
} SceCtrlData2;
```

#### **Members**

timeStamp Time stamp in the Controller Service (process time: μ sec) buttons Button state information \*Includes SCE\_CTRL\_L1, SCE\_CTRL\_R1, SCE\_CTRL\_L2, SCE\_CTRL\_R2, SCE\_CTRL\_L3, and SCE\_CTRL R3 (excludes SCE CTRL L and SCE CTRL R) Left stick X axis (left 0x00 to right 0xff) 1x 1 y Left stick Y axis (top 0x00 to bottom 0xff) Right stick X axis (left 0x00 to right 0xff) rх Right stick Y axis (top 0x00 to bottom 0xff) ry ир Pressure sensitivity value of the up key (disabled) Pressure sensitivity value of the right key (disabled) right down Pressure sensitivity value of the down key (disabled) left. Pressure sensitivity value of the left key (disabled) 12 Pressure sensitivity value of the L2 button r2 Pressure sensitivity value of the R2 button 11 Pressure sensitivity value of the L1 button (disabled) r1 Pressure sensitivity value of the R1 button (disabled) Pressure sensitivity value of the triangle button (disabled) triangle circle Pressure sensitivity value of the circle button (disabled) cross Pressure sensitivity value of the cross button (disabled) square Pressure sensitivity value of the square button (disabled) rsrv Reserved area

#### **Description**

This structure is for obtaining controller state information.

The timeStamp member stores the timestamp (process time) at which sampling was performed and data was obtained.

The *buttons* member stores 32-bit button state information. An application can use basic button state constants and SCE\_CTRL\_INTERCEPTED to determine the pressed/not pressed state of each button, etc.

The 1x, 1y, rx, and ry members store coordinate data of the analog sticks with 0x80 as the center.

The 12 and r2 members store pressure sensitive values from 0x00 to 0xFF.

Values for the up, right, down, left, 11, r1, triangle, circle, cross, and square are disabled. Obtain each button's state from the buttons member.

#### See Also

Basic Button State Constants, SCE\_CTRL\_INTERCEPTED, sceCtrlPeekBufferPositive2(), sceCtrlPeekBufferNegative2(), sceCtrlReadBufferPositive2(), sceCtrlReadBufferNegative2()

# **SceCtrlWirelessControllerInfo**

#### Wireless controller information

#### **Definition**

#### **Members**

connected Connection state reserved Reserved area

#### **Description**

This structure is for storing wireless controller information.

For the *connected* member, the connection state of the wireless controller will be stored.

Values will be stored in an array as follows for connected.

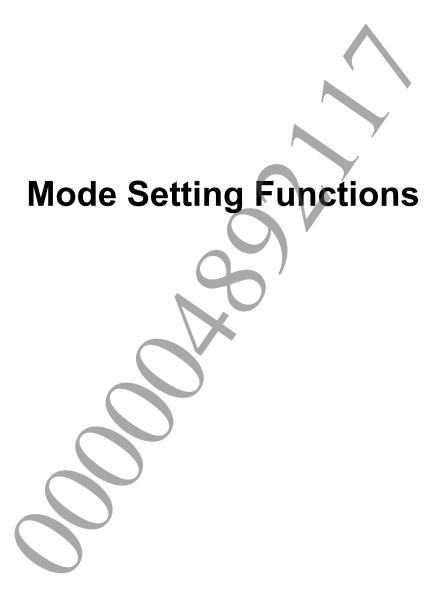
Member	Description
connected[0]	Connection state of the Controller Number 1 wireless controller
connected[1]	Connection state of the Controller Number 2 wireless controller
connected[2]	Connection state of the Controller Number 3 wireless controller
connected[3]	Connection state of the Controller Number 4 wireless controller

The maximum number of wireless controllers that can be connected is four.

#### See Also

sceCtrlGetWirelessControllerInfo()





# sceCtrlSetSamplingMode

#### Setting the controller mode

#### **Definition**

```
#include <ctrl.h>
int sceCtrlSetSamplingMode (
        SceUInt32 uiMode
);
```

#### **Arguments**

Controller mode uiMode

#### Return value

Returns the old mode value which had been specified up to that point for normal termination.

If an error occurs, a negative value is returned.

For details on the error, refer to "Error Codes."

#### **Description**

This function sets the controller mode.

Set the controller mode constant to the uiMode argument.

The default is buttons only mode (SCE CTRL MODE DIGITALONLY).

#### See Also

 $\verb|sceCtrlGetSamplingMode()|, Controller Mode Constants|$ 



# sceCtrlGetSamplingMode

#### Get controller mode

#### **Definition**

#### **Arguments**

puiMode Pointer to the SceUInt32 variable which is to receive the controller mode

#### Return value

Returns the controller mode setting value that is currently set for normal termination.

If an error occurs, a negative value is returned.

For details on the error, refer to "Error Codes."

#### **Description**

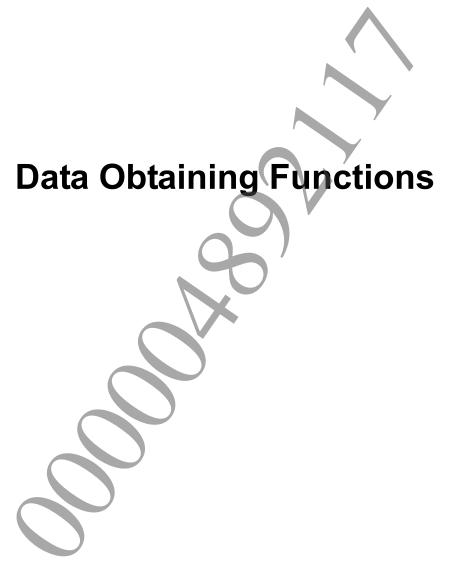
This function obtains the controller mode setting value that is currently set.

The setting can be changed with the sceCtrlSetSamplingMode() function.

#### See Also

sceCtrlSetSamplingMode(), Controller Mode Constants





# sceCtrlPeekBufferPositive, sceCtrlPeekBufferNegative

Obtain controller state information by polling

#### **Definition**

#### **Arguments**

port Type of data to obtain

pData Buffers to receive controller data

*nBufs* Number of buffers to receive controller data (1 to 64)

#### Return value

Stores the obtained controller data in *pData* and returns the number of obtained data for normal termination. The value is in the range of 1 to *nBufs*.

If an error occurs, a negative value is returned.

For details on the error, refer to "Error Codes."

#### Description

These functions obtain controller state information.

Each bit of the buttons member of the SceCtrlData structure to be stored in the pData argument will be as follows when the corresponding button is pressed.

- sceCtrlPeekBufferPositive():1 (positive logic)
- sceCtrlPeekBufferNegative():0 (negative logic)

The buttons member of the SceCtrlData structure will contain SCE\_CTRL\_L and SCE\_CTRL\_R (it will not contain SCE\_CTRL\_L1, SCE\_CTRL\_R1, SCE\_CTRL\_L2, SCE\_CTRL\_R2, SCE\_CTRL\_L3, and SCE\_CTRL\_R3).

The most recent 64 sets of controller information are stored in buffers of the Controller Service and sceCtrlPeekBufferPositive()/sceCtrlPeekBufferNegative() obtains the controller information of the number specified in the nBufs argument from the newest buffers by polling.

Buffer contents are updated by an interrupt every time the controller is sampled. Because of this, depending on the timing at which

sceCtrlPeekBufferPositive()/sceCtrlPeekBufferNegative() is called, the data that is obtained may differ in terms of being data before/after buffer updates (usually carried out at the VSYNC period).

**©SCEI** 

When using this function for PlayStation®Vita, specify 0 to port.

When using this function for PlayStation®TV, specify one of the following values to port.

port	Description	
0	Obtains merged data of the Controller Number 1 wireless controller, BD remote control, and a	
	TV remote control from HDMI CEC remote passthrough.	
1	Obtains data of the Controller Number 1 wireless controller.	
2	Obtains data of the Controller Number 2 wireless controller.	
3	Obtains data of the Controller Number 3 wireless controller.	
4	Obtains data of the Controller Number 4 wireless controller.	

Data obtained when port=0 is the merged value of controller and remote control inputs. It is recommended that 1 be specified to port when obtaining information of a single wireless controller.

#### **Notes**

The similar functions of sceCtrlReadBufferPositive() and sceCtrlReadBufferNegative() obtain controller state information by blocking; when calling either of these functions earlier than the sampling interval, the function will block the thread to obtain the most recent data.

#### See Also

SceCtrlData

# sceCtrlReadBufferPositive, sceCtrlReadBufferNegative

Obtain controller state information by blocking

#### **Definition**

#### **Arguments**

port Type of data to obtain

pData Buffers to receive controller data

*nBufs* Number of buffers to receive controller data (1 to 64)

#### Return value

Stores the obtained controller data in *pData* and returns the number of obtained data for normal termination. The value is in the range of 1 to *nBufs*.

If an error occurs, a negative value is returned.

For details on the error, refer to "Error Codes."

#### Description

These functions obtain controller state information.

Each bit of the buttons member of the SceCtrlData structure to be stored in the pData argument will be as follows when the corresponding button is pressed.

- sceCtrlReadBufferPositive():1 (positive logic)
- sceCtrlReadBufferNegative():0 (negative logic)

The buttons member of the SceCtrlData structure will contain SCE\_CTRL\_L and SCE\_CTRL\_R (it will not contain SCE\_CTRL\_L1, SCE\_CTRL\_R1, SCE\_CTRL\_L2, SCE\_CTRL\_R2, SCE\_CTRL\_L3, and SCE\_CTRL\_R3).

The Controller Service holds the most recent 64 sets of controller state information at maximum in its buffers. sceCtrlReadBufferPositive()/sceCtrlReadBufferNegative() obtains the controller information of the number specified in the nBufs argument the position of the current read pointer and then updates the read pointer..

If the controller was not sampled even once during the period from the last call of the sceCtrlReadBufferPositive()/sceCtrlReadBufferNegative() function until the current call of the sceCtrlReadBufferPositive()/sceCtrlReadBufferNegative() function, the thread is blocked in the sceCtrlReadBufferPositive()/sceCtrlReadBufferNegative() function until the controller is sampled again. Consequently, the return value will always be 1 or greater, and the most recent data will be obtained.

In addition, when the sceCtrlReadBufferPositive()/sceCtrlReadBufferNegative() function cannot be called at each specified sampling interval due to a processing lag (for example), set the value of nBuf to 2 or greater; based on whether or not the return value is greater than 1, it will be possible to determine whether or not there was a processing lag.

When using this function for PlayStation®Vita, specify 0 to port.

When using this function for PlayStation®TV, specify one of the following values to port.

port	Description	
0	Obtains merged data of the Controller Number 1 wireless controller, BD remote control, and a	
	TV remote control from HDMI CEC remote passthrough.	
1	Obtains data of the Controller Number 1 wireless controller.	
2	Obtains data of the Controller Number 2 wireless controller.	
3	Obtains data of the Controller Number 3 wireless controller.	
4	Obtains data of the Controller Number 4 wireless controller.	

Data obtained when port=0 is the merged value of controller and remote control inputs. It is recommended that 1 be specified to port when obtaining information of a single wireless controller.

#### **Notes**

The similar functions of sceCtrlPeekBufferPositive() and sceCtrlPeekBufferNegative() obtain controller state information by snooping; when calling either of these functions earlier than the sampling interval, the function will not block the thread and the last sampling result will be repeatedly obtained.

#### See Also

SceCtrlData



# sceCtrlPeekBufferPositive2, sceCtrlPeekBufferNegative2

Obtain controller state information by polling

#### **Definition**

#### **Arguments**

port Type of data to obtain

pData Buffers to receive controller data

Number of buffers to receive controller data (1 to 64)

#### **Return Values**

Stores the obtained controller data in *pData* and returns the number of obtained data for normal termination. The value is in the range of 1 to *nBufs*.

If an error occurs, a negative value is returned.

For details on the error, refer to "Error Codes."

#### Description

These functions obtain controller state information.

Each bit of the buttons member of the SceCtrlData2 structure to be stored in the pData argument will be as follows when the corresponding button is pressed.

- sceCtrlPeekBufferPositive2():1 (positive logic)
- sceCtrlPeekBufferNegative2(): 0 (negative logic)

The buttons member of the SceCtrlData2 structure will contain SCE\_CTRL\_L1, SCE\_CTRL\_R1, SCE\_CTRL\_L2, SCE\_CTRL\_R2, SCE\_CTRL\_L3, and SCE\_CTRL\_R3 (it will not contain SCE\_CTRL\_L and SCE\_CTRL\_R).

The Controller Service holds the most recent 64 sets of controller state information at maximum in its buffers. sceCtrlPeekBufferPositive2()/sceCtrlPeekBufferNegative2() obtains the controller information of the number specified in the nBufs argument from the newest buffers by polling.

Buffer contents are updated by an interrupt every time the controller is sampled. Because of this, depending on the timing at which

sceCtrlPeekBufferPositive2()/sceCtrlPeekBufferNegative2() is called, the data that is obtained may differ in terms of being data before/after buffer updates (usually carried out at the VSYNC period).

When using this function for PlayStation®Vita, specify 0 to port.

When using this function for PlayStation®TV, specify one of the following values to port.

port	Description	
0	Obtains merged data of the Controller Number 1 wireless controller, BD remote control, and a	
	TV remote control from HDMI CEC remote passthrough.	
1	Obtains data of the Controller Number 1 wireless controller.	
2	Obtains data of the Controller Number 2 wireless controller.	
3	Obtains data of the Controller Number 3 wireless controller.	
4	Obtains data of the Controller Number 4 wireless controller.	

Data obtained when port=0 is the merged value of controller and remote control inputs. It is recommended that 1 be specified to port when obtaining information of a single wireless controller.

#### **Notes**

The similar functions of  ${\tt sceCtrlReadBufferPositive2}$  () and

sceCtrlReadBufferNegative2() obtain controller state information by blocking; when calling either of these functions earlier than the sampling interval, the function will block the thread to obtain the most recent data.

#### See Also

SceCtrlData2

# sceCtrlReadBufferPositive2, sceCtrlReadBufferNegative2

Obtain controller state information by blocking

#### **Definition**

#### **Arguments**

Type of data to obtain

pData Buffers to receive controller data

nBufs Number of buffers to receive controller data (1 to 64)

#### **Return Values**

Stores the obtained controller data in pData and returns the number of obtained data for normal termination. The value is in the range of 1 to nBufs.

If an error occurs, a negative value is returned.

For details on the error, refer to "Error Codes."

#### Description

These functions obtain controller state information.

Each bit of the buttons member of the SceCtrlData2 structure to be stored in the pData argument will be as follows when the corresponding button is pressed.

- sceCtrlReadBufferPositive2():1 (positive logic)
- sceCtrlReadBufferNegative2():0 (negative logic)

The buttons member of the SceCtrlData2 structure will contain SCE\_CTRL\_L1, SCE\_CTRL\_R1, SCE\_CTRL\_L2, SCE\_CTRL\_R2, SCE\_CTRL\_L3, and SCE\_CTRL\_R3 (it will not contain SCE\_CTRL\_L and SCE\_CTRL\_R).

The most recent 64 sets of controller information are stored in buffers of the Controller Service and sceCtrlReadBufferPositive2()/sceCtrlReadBufferNegative2() obtains the controller information of the number specified in the nBufs argument from the position of the current read pointer and then updates the read pointer.

If the controller was not sampled even once during the period from the last call of the sceCtrlReadBufferPositive2()/sceCtrlReadBufferNegative2() function until the current call of the sceCtrlReadBufferPositive2()/sceCtrlReadBufferNegative2() function, the thread is blocked within the

sceCtrlReadBufferPositive2()/sceCtrlReadBufferNegative2() function until the controller is sampled again. Consequently, the return value will always be 1 or greater, and the most recent data will be obtained.

In addition, when the sceCtrlReadBufferPositive2()/sceCtrlReadBufferNegative2() function cannot be called at the set sampling interval due to a processing lag (for example), set the value of nBuf to 2 or greater; based on whether or not the return value is greater than 1, it will be possible to determine whether or not there was a processing lag.

When using this function for PlayStation®Vita, specify 0 to port.

When using this function for PlayStation®TV, specify one of the following values to port.

port	Description	
0	Obtains merged data of the Controller Number 1 wireless controller, BD remote control, and a	
	TV remote control from HDMI CEC remote passthrough.	
1	Obtains data of the Controller Number 1 wireless controller.	
2	Obtains data of the Controller Number 2 wireless controller.	
3	Obtains data of the Controller Number 3 wireless controller.	
4	Obtains data of the Controller Number 4 wireless controller.	

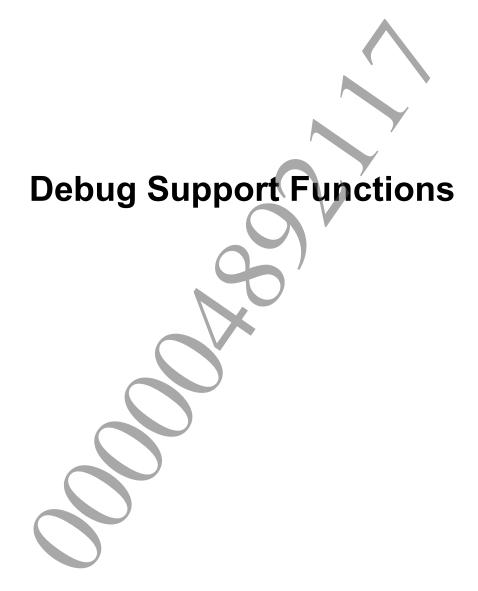
Data obtained when port=0 is the merged value of controller and remote control inputs. It is recommended that 1 be specified to port when obtaining information of a single wireless controller.

#### **Notes**

The similar functions of sceCtrlPeekBufferPositive2() and sceCtrlPeekBufferNegative2() obtain controller state information by snooping; when calling either of these functions earlier than the sampling interval, the function will not block the thread and the last sampling result will be repeatedly obtained.

#### See Also

SceCtrlData2



# Document serial number: 000004892117

# sceCtrlClearRapidFire

#### Clear rapid-fire settings

#### **Definition**

#### **Arguments**

port Specify 0.

*idx* Rule index for button rapid-fire (0 to 15)

#### Return value

Returns SCE OK(0) for normal termination.

If an error occurs, a negative value is returned.

For details on the error, refer to "Error Codes."

#### **Description**

This function clears the button rapid-fire settings which were set using sceCtrlSetRapidFire(). In the idx argument, select the rule index to clear.

#### See Also

sceCtrlSetRapidFire()



# sceCtrlSetRapidFire

#### Set button rapid-fire rules

#### **Definition**

#### **Arguments**

port Specify 0. idx Rule index for button rapid-fire (0 to 15) pRule Rapid-fire trigger rules

#### Return value

Returns SCE OK (0) for normal termination.

If an error occurs, a negative value is returned.

For details on the error, refer to the section "Error Codes."

#### **Description**

This function sets the rules for the button rapid-fire functionality.

It will work on the retail units as well, but is primarily intended for debugging.

Place a rule index number from 0 to 15 in the *idx* argument. Up to 16 patterns can be stored as rules.

The button rapid-fire rules are set with the structure specified in the pRule argument.

For the set rules, when the value obtained by masking the button input with <code>pRule->uiMask</code> matches <code>pRule->uiTrigger</code>, the button specified with <code>pRule->uiTarget</code> is placed in the rapid-fire state. <code>pRule->uiMask</code>, <code>pRule->uiTrigger</code> and <code>pRule->uiTarget</code> all specify the logic in positive logic. The rapid-fire start timing is specified with the <code>pRule->uiDelay</code> argument, and the rapid-fire cycle is specified with <code>pRule->uiMake</code> and <code>pRule->uiBreak</code>.



#### **Sample Settings**

```
SceCtrlRapidFireRule rule1, rule2;
//rule1
                 = SCE CTRL CIRCLE | SCE CTRL L; // Mask circle and L buttons
rule1.uiMask
rule1.uiTrigger = SCE CTRL CIRCLE;
                                                    // If the circle button is in
the pressed state
                                                     // Using the square button,
rule1.uiTarget = SCE CTRL SQUARE;
rule1.uiDelay
                                                     // after a 10-cycle delay,
                 = 10;
rule1.uiMake
                 = 1:
                                                    // with a (1+1)-cycle period:
ON \rightarrow OFF \rightarrow
rule1.uiBreak
                                                     //
                 = 1;
//Set rule 1 to index 0
sceCtrlSetRapidFire(0,
                                // port=0
                      Ο,
                                // idx = 0
                      &rule1); // rule1
//rule2
rule2.uiMask
                 = SCE CTRL TRIANGLE; // Mask triangle button
rule2.uiTrigger = SCE CTRL TRIANGLE; // If the triangle button is in the pressed
rule2.uiTarget
                 = SCE CTRL TRIANGLE; // Using the triangle button,
rule2.uiDelay
                                            after a 1-cycle delay,
                 = 1;
rule2.uiMake
                 = 2;
                                            with a (2+1) cycle period: ON\rightarrowON\rightarrowOFF \rightarrow
rule2.uiBreak
                 = 1;
//Set rule 2 to index 1
sceCtrlSetRapidFire(0,
                                //port
                      1,
                                //idx
                      &rule2); //rule
```

#### **Notes**

In this function, an error (SCE\_CTRL\_ERROR\_INVALID\_ARG) will occur if any one of the uiDelay, uiMake and uiBreak members of the SceCtrlRapidFireRule structure is set to 64 or greater. Because the maximum number of data buffers in the Controller Service is 64, setting rules exceeding this number of buffers is not allowed.

Furthermore, it is also prohibited to set 0 in all of uiDelay, uiMake and uiBreak because such setting does not make sense.

#### See Also

sceCtrlClearRapidFire()



# sceCtrllsMultiControllerSupported

Get multiple wireless controller support state

#### **Definition**

#include <ctrl.h> SceBool sceCtrlIsMultiControllerSupported(void);

#### **Arguments**

None

#### **Return Values**

Returns true when multiple wireless controllers are supported and false when not supported.

#### **Description**

This function obtains whether or not the environment supports multiple wireless controllers.

# sceCtrlGetWirelessControllerInfo

Get wireless controller information

#### **Definition**

```
#include <ctrl.h>
int sceCtrlGetWirelessControllerInfo (
        SceCtrlWirelessControllerInfo* pInfo
);
```

#### **Arguments**

pInfo Buffer to receive wireless controller information

#### **Return Values**

Returns SCE OK (0) for normal termination. If an error occurs, a negative value is returned. For details on the error, refer to the section "Error Codes."

#### **Description**

This function obtains information of the wireless controller. For details on wireless controller information, refer to the description for SceCtrlWirelessControllerInfo.

#### See Also

SceCtrlWirelessControllerInfo, Wireless Controller Connection State Constants

