libdbg Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

Data Types

SceDbgBreakOnErrorState

An enumeration to represent the various states for “break on error” functionality.

Definition

```
#include <libdbg.h>
typedef enum SceDbgBreakOnErrorState {
        SCE_DBG_DISABLE_BREAK_ON_ERROR = 0,
        SCE_DBG_ENABLE_BREAK_ON_ERROR
} SceDbgBreakOnErrorState;
```

Enumeration Values

Macro	Description
`SCE_DBG_DISABLE_BREAK_ON_ERROR`	The library will not break execution after outputting an error.
`SCE_DBG_ENABLE_BREAK_ON_ERROR`	The library will break execution after outputting an error.

Description

An enumeration to represent the various states for “break on error” functionality.

See Also

`sceDbgSetBreakOnErrorState`

SceDbgLogLevel

An enumeration to represent the various logging levels which can be output by sceDbgLoggingHandler.

Definition

```
#include <libdbg.h>
typedef enum SceDbgLogLevel {
        SCE_DBG_LOG_LEVEL_TRACE = 0,
        SCE_DBG_LOG_LEVEL_DEBUG,
        SCE_DBG_LOG_LEVEL_INFO,
        SCE_DBG_LOG_LEVEL_WARNING,
        SCE_DBG_LOG_LEVEL_ERROR,
        SCE_DBG_NUM_LOG_LEVELS
} SceDbgLogLevel;
```

Enumeration Values

Macro	Description
SCE_DBG_LOG_LEVEL_TRACE	An extremely verbose logging level, mostly useful for internal developers.
SCE_DBG_LOG_LEVEL_DEBUG	A diagnostic logging level.
SCE_DBG_LOG_LEVEL_INFO	An informational logging level.
SCE_DBG_LOG_LEVEL_WARNING	A logging level that gives warnings of situations detrimental to proper execution.
SCE_DBG_LOG_LEVEL_ERROR	A logging level that will report erroneous conditions in execution.
SCE_DBG_NUM_LOG_LEVELS	The number of logging levels available.

Description

An enumeration to represent the various logging levels which can be output by sceDbgLoggingHandler.

See Also

sceDbgSetMinimumLogLevel, sceDbgLoggingHandler

Functions

sceDbgAssertionHandler

Outputs a message via the assertion handler.

Definition

```
#include <libdbg.h>
SceInt32 sceDbgAssertionHandler(
        const char *pFile,
        int line,
        bool stop,
        const char *pComponent,
        const char *pMessage,
        ...
);
```

Arguments

[in] *pFile*	The file name at which the assert originated.
[in] *line*	The line number within the file at which the assert originated.
[in] *stop*	A flag that indicates to the caller whether the program should stop execution in the event of an assertion. The caller receives this flag as a return value and must interpret it as appropriate.
[in] *pComponent*	An identifier for the component from which the assert originated (e.g. "libGxm" or "Physics").
[in] *pMessage*	A format string in the "`printf-style`" for the output message (e.g. "The binding named: `%s` has an invalid value: `%d`").
[in] `...`	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Return Values

Value	Description
`SCE_OK`	The assert did not fire and the operation completed successfully.
stop	In the event that the assert fired the caller-specified value *stop* is returned.

Description

Outputs a message via the assertion handler. Messages are limited to a total length of 512 characters. Messages greater than this length will be truncated.

sceDbgLoggingHandler

Outputs a message via the logging handler.

Definition

```
#include <libdbg.h>
SceInt32 sceDbgLoggingHandler(
        const char *pFile,
        int line,
        int severity,
        const char *pComponent,
        const char *pMessage,
        ...
);
```

Arguments

[in] *pFile*	The file name at which the assert originated.
[in] *line*	The line number within the file at which the assert originated.
[in] *severity*	The severity of the message. Only messages with a severity greater than or equal to that set using `sceDbgSetMinimumLogLevel()` will be output to TTY.
[in] *pComponent*	An identifier for the component from which the assert originated (e.g. “libGxm” or “Physics”).
[in] *pMessage*	A format string in the “`printf-style`” for the output message (e.g. “The binding named: `%s` has an invalid value: `%d`”).
[in] `...`	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Return Values

Value	Description
`SCE_OK`	The operation was completed successfully.

Description

Outputs a message via the logging handler. Messages are limited to a total length of 512 characters. Messages greater than this length will be truncated and an error returned.

sceDbgSetBreakOnErrorState

Specifies whether the library should break execution when a client library outputs an error.

Definition

```
#include <libdbg.h>
SceInt32 sceDbgSetBreakOnErrorState(
        SceDbgBreakOnErrorState state
);
```

Arguments

[in] *state*	An enum value specifying whether or not the library should break after outputting an error.

Return Values

Value	Description
SCE_OK	The operation completed successfully.

Description

Specifies whether the library should break execution when a client library outputs an error. The default setting is SCE_DBG_DISABLE_BREAK_ON_ERROR (i.e. execution will not break on error).

sceDbgSetMinimumLogLevel

Specifies the minimum severity level for the output of logging information.

Definition

```
#include <libdbg.h>
SceInt32 sceDbgSetMinimumLogLevel(
        SceInt32 minimumLogLevel
);
```

Arguments

[in] *minimumLogLevel*	The minimum severity at which debugging messages should be output.

Return Values

Value	Description
`SCE_OK`	The operation was completed successfully.

Description

Specifies the minimum severity level for the output of logging information. The default level is `SCE_DBG_LOG_LEVEL_TRACE`.

Macros

SCE_BREAK

Breaks program execution.

Definition

```
#include <libdbg.h>
#define SCE_BREAK() _SCE_BREAK()
```

Arguments

None

Description

Breaks program execution. If a debugger is attached, the user can resume execution immediately.

SCE_DBG_ALWAYS_ASSERT

An always assert macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_ALWAYS_ASSERT(
        test
) SCE_DBG_ASSERT_PRIVATE(test, true, SCE_BREAK(), "Assertion
failed: %s\n",#test)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.

Description

An always assert macro. This will always output a simple message and halt execution if the condition specified by *test* evaluates to false.

Notes

If the condition evaluates to false, this assert will always fire regardless of the value of `SCE_DBG_ASSERTS_ENABLED`.

SCE_DBG_ALWAYS_ASSERT_EQUAL

An always assert equal macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_ALWAYS_ASSERT_EQUAL(
        value,
        expected
) SCE_DBG_ASSERT_PRIVATE((value == expected), true, SCE_BREAK(), "Assertion
failed, values not equal\n")
```

Arguments

value	An identifier to evaluate for equality. This is assumed to be equal to the *expected* parameter under normal circumstances.
expected	An identifier to evaluate for equality. This denotes the expected value and is assumed to be equal to the *value* parameter under normal circumstances.

Description

An always assert equal macro. This will always output a simple message and halt execution if the *value* and *expected* parameters are not equal.

Notes

If the condition evaluates to false, this assert will always fire regardless of the value of `SCE_DBG_ASSERTS_ENABLED`. The *value* and *expected* parameters must be valid operands to the binary == operator.

SCE_DBG_ALWAYS_ASSERT_MSG

An always assert message macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_ALWAYS_ASSERT_MSG(
        test,
        msg,
        ...
) SCE_DBG_ASSERT_PRIVATE(test, true, SCE_BREAK(), msg, ##__VA_ARGS__)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.
msg	A message string in the "`printf-style`" for the output message (e.g. "The binding named: `%s` has an invalid value: `%d`").
...	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

An always assert message macro. A user-supplied message will be output and execution will always be halted if the condition specified by *test* evaluates to false.

Notes

If the condition evaluates to false, this assert will always fire regardless of the value of `SCE_DBG_ASSERTS_ENABLED`.

SCE_DBG_ASSERT

An assert macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_ASSERT(
        test
) SCE_DBG_ASSERT_PRIVATE(test, true, SCE_BREAK(), "Assertion failed: %s\n",
#test)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.

Description

An assert macro. A simple message will be output and execution halted if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the condition specified by *test* evaluates to false.

Notes

This assert will be removed at compile-time if the value of `SCE_DBG_ASSERTS_ENABLED` is zero. The eventual intention is for the user to be able to resume immediately in the event of this assert firing.

SCE_DBG_ASSERT_EQUAL

An assert equal macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_ASSERT_EQUAL(
        value,
        expected
) SCE_DBG_ASSERT_PRIVATE(((value) == (expected)), true, SCE_BREAK(), "Assertion
failed, values not equal\n")
```

Arguments

value	An identifier to evaluate for equality. This is assumed to be equal to the *expected* parameter under normal circumstances.
expected	An identifier to evaluate for equality. This denotes the expected value and is assumed to be equal to the *value* parameter under normal circumstances.

Description

An assert equal macro. A simple message will be output and execution halted if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the *value* and *expected* parameters are not equal.

Notes

This assert will be removed at compile-time if the value of `SCE_DBG_ASSERTS_ENABLED` is zero. The eventual intention is for the user to be able to resume immediately in the event of this assert firing. The *value* and *expected* parameters must be valid operands to the binary == operator.

SCE_DBG_ASSERT_MSG

An assert message macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_ASSERT_MSG(
        test,
        msg,
        ...
) SCE_DBG_ASSERT_PRIVATE(test, true, SCE_BREAK(), msg, ##__VA_ARGS__)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.
msg	A message string in the "printf-style" for the output message. (e.g. "The binding named: `%s` has an invalid value: `%d`").
...	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

An assert message macro. A user-supplied message will be output and execution halted if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the condition specified by *test* evaluates to false.

Notes

This assert will be removed at compile-time if the value of `SCE_DBG_ASSERTS_ENABLED` is zero.

SCE_DBG_LOG_DEBUG

Outputs a debug message via the logging handler.

Definition

```
#include <libdbg.h>
#define SCE_DBG_LOG_DEBUG(
        format,
        ...
) SCE_DBG_LOG_BASE(SCE_DBG_LOG_LEVEL_DEBUG, SCE_DBG_LOG_COMPONENT, format,
##__VA_ARGS__)
```

Arguments

format	A format string in the "printf-style" for the output message (e.g. "The binding named: %s has an invalid value: %d").
...	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

Outputs a debug message via the logging handler.

Notes

The message is only output to TTY if both the compile time minimum log-level SCE_DBG_MINIMUM_LOG_LEVEL and the runtime minimum log-level (set using sceDbgSetMinimumLogLevel()) are less than or equal to SCE_DBG_LOG_LEVEL_DEBUG.

SCE_DBG_LOG_ERROR

Outputs an error message via the logging handler.

Definition

```
#include <libdbg.h>
#define SCE_DBG_LOG_ERROR(
        format,
        ...
) SCE_DBG_LOG_BASE(SCE_DBG_LOG_LEVEL_ERROR, SCE_DBG_LOG_COMPONENT, format,
##__VA_ARGS__)
```

Arguments

format	A format string in the "printf-style" for the output message (e.g. "The binding named: `%s` has an invalid value: `%d`").
`...`	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

Outputs an error message via the logging handler.

Notes

The message is only output to TTY if both the compile time minimum log-level `SCE_DBG_MINIMUM_LOG_LEVEL` and the runtime minimum log-level (set using `sceDbgSetMinimumLogLevel()`) are less than or equal to `SCE_DBG_LOG_LEVEL_ERROR`.
An optional component identifier can be specified prior to usage with: `SCE_DBG_LOG_COMPONENT`.

SCE_DBG_LOG_INFO

Outputs an info message via the logging handler.

Definition

```
#include <libdbg.h>
#define SCE_DBG_LOG_INFO(
        format,
        ...
) SCE_DBG_LOG_BASE(SCE_DBG_LOG_LEVEL_INFO, SCE_DBG_LOG_COMPONENT, format,
##__VA_ARGS__)
```

Arguments

format	A format string in the "printf-style" for the output message (e.g. "The binding named: %s has an invalid value: %d").
...	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

Outputs an info message via the logging handler.

Notes

The message is only output to TTY if both the compile time minimum log-level SCE_DBG_MINIMUM_LOG_LEVEL and the runtime minimum log-level (set using sceDbgSetMinimumLogLevel()) are less than or equal to SCE_DBG_LOG_LEVEL_INFO.

SCE_DBG_LOG_TRACE

Outputs a trace message via the logging handler.

Definition

```
#include <libdbg.h>
#define SCE_DBG_LOG_TRACE(
        format,
        ...
) SCE_DBG_LOG_BASE(SCE_DBG_LOG_LEVEL_TRACE, SCE_DBG_LOG_COMPONENT, format,
##__VA_ARGS__)
```

Arguments

format	A format string in the "printf-style" for the output message (e.g. "The binding named: `%s` has an invalid value: `%d`").
`...`	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

Outputs a trace message via the logging handler.

Notes

The message is only output to TTY if both the compile time minimum log-level `SCE_DBG_MINIMUM_LOG_LEVEL` and the runtime minimum log-level (set using `sceDbgSetMinimumLogLevel()`) are less than or equal to `SCE_DBG_LOG_LEVEL_TRACE`.

SCE_DBG_LOG_WARNING

Outputs a warning message via the logging handler.

Definition

```
#include <libdbg.h>
#define SCE_DBG_LOG_WARNING(
        format,
        ...
) SCE_DBG_LOG_BASE(SCE_DBG_LOG_LEVEL_WARNING, SCE_DBG_LOG_COMPONENT, format,
##__VA_ARGS__)
```

Arguments

format	A format string in the “printf-style” for the output message (e.g. “The binding named: %s has an invalid value: %d”).
...	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

Outputs a warning message via the logging handler.

Notes

The message is only output to TTY if both the compile time minimum log-level SCE_DBG_MINIMUM_LOG_LEVEL and the runtime minimum log-level (set using sceDbgSetMinimumLogLevel()) are less than or equal to SCE_DBG_LOG_LEVEL_WARNING.

SCE_DBG_SIMPLE_ASSERT

An assert macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_SIMPLE_ASSERT(
        test
) _SCE_MACRO_BEGIN \
        if (SCE_UNLIKELY(!(test))) { \
        SCE_BREAK(); \
        } \
        _SCE_MACRO_END
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.

Description

An assert macro. Execution will be halted if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the condition specified by *`test`* evaluates to false.

Notes

This assert will be removed at compile-time if the value of `SCE_DBG_ASSERTS_ENABLED` is zero. The eventual intention is for the user to be able to resume immediately in the event of this assert firing.

SCE_DBG_STATIC_ASSERT

A static (compile-time) assertion macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_STATIC_ASSERT(
        condition
) enum { \
        SCE_DBG_CONCATENATE(SCE_DBG_STATIC_ASSERT_ENUM_, __LINE__) \
        = sizeof(SCE_DBG_STATIC_ASSERT_FAILED< (bool)(condition) >) \
        }
```

Arguments

condition	A test condition to evaluate. This is assumed to be true under normal circumstances.

Description

A static (compile-time) assertion macro. This will produce a compile-compilation error if the condition specified by *condition* evaluates to false.

SCE_DBG_STOP_ASSERT

An assert macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_STOP_ASSERT(
        test
) SCE_DBG_ASSERT_PRIVATE(test, true, SCE_STOP(), "Assertion failed: %s\n",
#test)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.

Description

An assert macro. A simple message will be output and execution halted if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the condition specified by *test* evaluates to false.

Notes

This assert will be removed at compile-time if the value of `SCE_DBG_ASSERTS_ENABLED` is zero. The eventual intention is that the user must manually move the program counter to be able to resume execution in the event of this assert firing.

SCE_DBG_VERIFY

A verify macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_VERIFY(
        test
) SCE_DBG_ASSERT_PRIVATE(test, true, SCE_BREAK(), "Assertion
failed: %s\n",#test)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.

Description

A verify macro. A simple message will be output and execution halted if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the condition specified by *`test`* evaluates to false.

Notes

The test condition will still be evaluated at compile-time even if `SCE_DBG_ASSERTS_ENABLED` is zero. However, in that case the assert will not fire. The eventual intention is for the user to be able to resume immediately in the event of this assert firing.

SCE_DBG_VERIFY_MSG

A verify message macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_VERIFY_MSG(
        test,
        msg,
        ...
) SCE_DBG_ASSERT_PRIVATE(test, true, SCE_BREAK(), msg, ##__VA_ARGS__)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.
msg	A message string in the "printf-style" for the output message (e.g. "The binding named: `%s` has an invalid value: `%d`").
`...`	A variable number of parameters, which will be inserted into the format string. The number and types of these parameters should match those specified in the format string.

Description

A verify message macro. A user-supplied message will be output and execution halted if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the condition specified by *`test`* evaluates to false.

Notes

The test condition will still be evaluated at compile-time even if `SCE_DBG_ASSERTS_ENABLED` is zero. However, in that case the assert will not fire. The eventual intention is for the user to be able to resume immediately in the event of this assert firing.

SCE_DBG_WARN_ASSERT

An assert macro.

Definition

```
#include <libdbg.h>
#define SCE_DBG_WARN_ASSERT(
        test
) SCE_DBG_ASSERT_PRIVATE(test, false, (void)0, "Warning - Assertion
failed: %s\n", #test)
```

Arguments

test	A test condition to evaluate. This is assumed to be true under normal circumstances.

Description

An assert macro. A warning message will be output if `SCE_DBG_ASSERTS_ENABLED` evaluates to true and the condition specified by *`test`* evaluates to false.

Notes

This assert will be removed at compile-time if the value of `SCE_DBG_ASSERTS_ENABLED` is zero. This assert will not result in a break in execution.

SCE_NORETURN_STOP

Stops program execution.

Definition

```
#include <libdbg.h>
#define SCE_NORETURN_STOP() _SCE_NORETURN_STOP()
```

Arguments

None

Description

Stops program execution. If a debugger is attached, the user must move the Program Counter (PC) before resuming execution.

Notes

Currently this macro behaves the same as `SCE_STOP`, but in a future release the compiler may generate code on the assumption that execution will not resume after reaching the resultant breakpoint.

SCE_STOP

Stops program execution.

Definition

```
#include <libdbg.h>
#define SCE_STOP() _SCE_STOP()
```

Arguments

None

Description

Stops program execution. If a debugger is attached, the user must move the Program Counter (PC) before resuming execution.

Defines

Define Summary

Define	Value	Description
`SCE_DBG_ASSERT_COMPONENT`	`""`	Optional component specifier for the `SCE_DBG_XXX` assertion macros.
`SCE_DBG_ASSERTS_ENABLED`	`0`	Optional compile-time flag to control whether or not the assert functionality provided using the `SCE_DBG_ASSERT_XXX` macros is enabled.
`SCE_DBG_LOG_COMPONENT`	`""`	Optional component specifier for the `SCE_DBG_LOG_XXX` macros.
`SCE_DBG_LOG_PREFIX`	`""`	Optional message prefix for the `SCE_DBG_LOG_XXX` macros.
`SCE_DBG_LOGGING_ENABLED`	`1`	Optional compile-time flag to control whether or not the logging functionality provided using the `SCE_DBG_LOG_XXX` macros is enabled. This will not affect prebuilt libraries or PRX.
`SCE_DBG_MINIMUM_LOG_LEVEL`	`SCE_DBG_LOG_LEVEL_TRACE`	Compile-time switch to control minimum log level output from the `SCE_DBG_LOG_XXX` macros. This can be used to remove logging entirely.