libult Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

User Level Thread Runtime

SceUltUlthreadRuntime

User level thread runtime structure

Definition

```
#include <ult.h>
typedef struct SceUltUlthreadRuntime{
        /* private */
} SceUltUlthreadRuntime;
```

Description

This is the user level thread runtime structure.

SceUltUlthreadRuntimeOptParam

User level thread runtime options

Definition

```
#include <ult.h>
typedef struct SceUltUlthreadRuntimeOptParam{
        uint32_t oneShotThreadStackSize;
        int32_t workerThreadPriority;
        uint32_t workerThreadCpuAffinityMask;
        uint32_t workerThreadAttr;
        const SceKernelThreadOptParam* workerThreadOptParam;
        /* Other members are private */
} SceUltUlthreadRuntimeOptParam;
```

Members

oneShotThreadStackSize	Stack size of one-shot thread (Default value: `SCE_KERNEL_THREAD_STACK_SIZE_MIN`)
workerThreadPriority	Priority of worker thread (Default value: `SCE_KERNEL_DEFAULT_PRIORITY_USER`)
workerThreadCpuAffinityMask	Affinity mask of worker thread (Default value: `SCE_KERNEL_CPU_MASK_USER_ALL`)
workerThreadAttr	Attribute of worker thread (Default value: 0)
workerThreadOptParam	Option of worker thread (Default value: NULL)

Description

This is the structure used to specify the options to be used during user level thread creation.

After executing `sceUltUlthreadRuntimeOptParamInitialize()` to initialize the structure, set values to the member variables for use.

oneShotThreadStackSize is the size of the stack used during one-shot thread execution. The stack for one-shot threads is allocated in the worker thread stack area.

workerThreadPriority, *workerThreadCpuAffinityMask*, *workerThreadAttr*, *workerThreadOptParam* are the worker thread attributes specified to `sceUltUlthreadRuntimeCreate()` during worker thread creation. For details, refer to `sceUltUlthreadRuntimeCreate()`.

The worker thread stack size is *oneShotThreadStackSize* + 4KiB.

sceUltUlthreadRuntimeOptParamInitialize

Initialize user level thread runtime options

Definition

```
#include <ult.h>
int32_t sceUltUlthreadRuntimeOptParamInitialize(
        SceUltUlthreadRuntimeOptParam *optParam
)
```

Arguments

optParam	Option structure of the user level thread runtime to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*optParam* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values for all the member variables.

sceUltUlthreadRuntimeGetWorkAreaSize

Get size of work area used for user level thread runtime

Definition

```
#include <ult.h>
uint32_t sceUltUlthreadRuntimeGetWorkAreaSize(
        uint32_t maxNumUlthread,
        uint32_t numWorkerThread
)
```

Arguments

maxNumUlthread	Maximum number of user level threads that can be created for this runtime
numWorkerThread	Number of worker threads

Return Values

Size of the work area to be used for user level thread runtime

Description

This function gets the buffer size required to create the user level thread runtime.

Prepare a buffer of the size obtained with this function and specify it to the *workArea* argument of `sceUltUlthreadRuntimeCreate()`.

For details about the arguments, refer to `sceUltUlthreadRuntimeCreate()`.

sceUltUlthreadRuntimeCreate

Create user level thread runtime

Definition

```
#include <ult.h>
int32_t sceUltUlthreadRuntimeCreate(
        SceUltUlthreadRuntime *runtime,
        const char* name,
        uint32_t maxNumUlthread,
        uint32_t numWorkerThread,
        void* workArea,
        SceUltUlthreadRuntimeOptParam *optParam
)
```

Arguments

runtime	Pointer to the user level thread runtime structure to be initialized
name	Name of runtime (for debugging). Only the first 31 characters are valid
maxNumUlthread	Maximum number of user level threads that can be created for this runtime
numWorkerThread	Number of worker threads
workArea	Work area of runtime
optParam	Runtime option If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*runtime*, *name*, or *workArea* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*runtime*, *workArea*, or *optParam* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	*maxNumUlthread* or *numWorkerThread* is 0. Or *optParam* is other than NULL and invalid
Other than above		Error during worker thread creation. Refer to `sceKernelCreateThread()` of the "Kernel Reference" document

Description

This function creates the user level thread runtime.

For *workArea*, specify the work buffer of the size obtained with `sceUltUlthreadRuntimeGetWorkAreaSize()`. Do not release this buffer until runtime is destroyed with `sceUltUlthreadRuntimeDestroy()`.

sceUltUlthreadRuntimeDestroy

Destroy user level thread runtime

Definition

```
#include <ult.h>
int32_t sceUltUlthreadRuntimeDestroy(
        SceUltUlthreadRuntime *runtime
)
```

Arguments

runtime	User level thread runtime structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*runtime* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*runtime* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	A user level thread for which `sceUltUlthreadJoin()` is not completed exists

Description

This function destroys the user level thread runtime.

SceUltUlthreadRuntimeInfo

User level thread runtime information structure

Definition

```
#include <ult.h>
typedef struct SceUltUlthreadRuntimeInfo{
        uint32_t size;
        uint32_t ulthreadRuntimeId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t numUlthread;
        uint32_t maxNumUlthread;
        uint32_t numWorkerThread;
        uint32_t oneShotThreadStackSize;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                              - (SCE_ULT_MAX_NAME_LENGTH+1)
                              - sizeof(uint32_t)*4)
                              / sizeof(uint64_t)];
} SceUltUlthreadRuntimeInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltUlthreadRuntimeInfo)`)
ulthreadRuntimeId	User level thread runtime ID
name	Name of user level thread runtime
numUlthread	Number of created user level threads
maxNumUlthread	Maximum number of user level threads that can be created
numWorkerThread	Number of worker threads
oneShotThreadStackSize	One-shot thread stack size
__reserved__	Reserved area

Description

This structure stores the user level thread runtime information by `sceUltGetUlthreadRuntimeInfo()`.

For *size*, always first assign `sizeof(SceUltUlthreadRuntimeInfo)` and then specify it for the function argument.

sceUltGetUlthreadRuntimeInfo

Get user level thread runtime status

Definition

```
#include <ult.h>
int32_t sceUltGetUlthreadRuntimeInfo(
        SceUltUlthreadRuntime *runtime,
        SceUltUlthreadRuntimeInfo *pInfo,
        SceUltUlthread **ulthreadList,
        const uint32_t maxNumUlthread,
        uint32_t *workerThreadIdList,
        const uint32_t maxNumWorkerThreadId
)
```

Arguments

runtime	Pointer to the user level thread runtime structure
pInfo	Pointer to the user level thread runtime information structure
ulthreadList	Pointer to the user level thread list or NULL
maxNumUlthread	Maximum number of user level threads to obtain
workerThreadIdList	Pointer to worker thread ID list or NULL
maxNumWorkerThreadId	Maximum number of worker thread IDs to obtain

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*runtime* or *pInfo* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*runtime* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains the user level thread runtime status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

User Level Threads

SceUltUlthread

User level thread structure

Definition

```
#include <ult.h>
typedef struct SceUltUlthread{
        /* private */
} SceUltUlthread;
```

Description

This is the user level thread structure.

SceUltUlthreadEntry

Entry Function of User level thread

Definition

```
#include <ult.h>
typedef int32_t (*SceUltUlthreadEntry)(uint32_t arg);
```

Description

Type of entry function of user level thread. For *arg* the argument for a user level thread is passed.

SceUltUlthreadOptParam

User level thread options

Definition

```
#include <ult.h>
typedef struct SceUltUlthreadOptParam{
        uint32_t attribute;
        /* Other members are private */
} SceUltUlthreadOptParam;
```

Members

attribute	Attribute of user level thread (Default value: 0)

Description

This structure is used to specify the options during user level thread creation.

After executing `sceUltUlthreadOptParamInitialize()` to initialize the structure, set values to the member variables for use.

Specify the bitwise OR of the following value to *attribute*. The default value is 0.

Value	(Number)	Description
SCE_ULT_ULTHREAD_ATTRIBUTE_FORCE_WAIT	0x1	This value enables one-shot threads, which normally cannot enter the wait state, to wait sync event by forcibly setting worker threads to the wait state when sync event wait is required. This value can be specified only for one-shot threads

sceUltUlthreadOptParamInitialize

Initialize user level thread options

Definition

```
#include <ult.h>
int32_t sceUltUlthreadOptParamInitialize(
        SceUltUlthreadOptParam *optParam
)
```

Arguments

`optParam`	Option structure of the user level thread to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*`optParam`* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*`optParam`* is not 8-byte aligned

Description

This function initializes *`optParam`* and sets the default values for all the member variables.

sceUltUlthreadCreate

Create user level thread

Definition

```
#include <ult.h>
int32_t sceUltUlthreadCreate(
        SceUltUlthread *ulthread,
        const char* name,
        SceUltUlthreadEntry entry,
        uint32_t arg,
        void* context,
        uint32_t sizeContext,
        SceUltUlthreadRuntime *runtime,
        SceUltUlthreadOptParam *optParam
)
```

Arguments

ulthread	Pointer to the user level thread structure to be initialized
name	Name of the thread (for debugging). Only the first 31 characters are valid
entry	Entry function
arg	Argument of the entry function
context	Thread execution context buffer
sizeContext	Size of execution context area (512 bytes minimum)
runtime	Runtime for executing the user level thread
optParam	Option. If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*ulthread*, *name*, *entry*, or *runtime* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*ulthread*, *context*, *optParam*, or *runtime* is not 8-byte aligned
`SCE_ULT_ERROR_RANGE`	0x80810003	*sizeContext* is greater than 0 and smaller than 512
`SCE_ULT_ERROR_INVALID`	0x80810004	*context* is other than NULL and *sizeContext* is 0 or not a multiple of 8. Also, *optParam* is other than NULL and invalid
`SCE_ULT_ERROR_STATE`	0x80810006	*runtime* has been destroyed
`SCE_ULT_ERROR_AGAIN`	0x80810008	The maximum number of user level threads that can be created for *runtime* already exists

Description

This function creates a user level thread. The created thread is executed by the worker threads of the specified runtime.

For *context*, specify the context buffer to be used for user level thread execution. This buffer is used for the thread stack and saving of registers during context switching.

A one-shot thread is created when NULL is specified for *context*. Since a one-shot thread does not have a context buffer that belongs to the thread, it is not possible to use an interface such that can enter the wait state.
One-shot threads are executed using dedicated stack areas assigned to worker threads.

sceUltUlthreadExit

Terminate user level thread

Definition

```
#include <ult.h>
int32_t sceUltUlthreadExit(
        int32_t status
);
```

Arguments

status	Exit code of thread

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_PERMISSION`	0x80810005	The caller is not a user level thread

Description

This function terminates the user level thread.

sceUltUlthreadYield

Yield worker thread

Definition

```
#include <ult.h>
int32_t sceUltUlthreadYield(void);
```

Arguments

None

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_PERMISSION`	0x80810005	The caller is not a user level thread. Or the function is called from a one-shot thread

Description

This function yields the worker thread that is currently executing the caller user level thread to another.

sceUltUlthreadGetSelf

Get user level thread structure of user level thread currently being executed

Definition

```
#include <ult.h>
int32_t sceUltUlthreadGetSelf(
        SceUltUlthread **ulthread
);
```

Arguments

`ulthread`	User level thread structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*`ulthread`* is NULL

Description

This function writes the pointer of the user level thread structure of the user level thread currently being executed to *`ulthread`*.

If the thread that called this function is not a user level thread, NULL is written to *`ulthread`*.

sceUltUlthreadJoin, sceUltUlthreadTryJoin

Wait termination of user level thread

Definition

```
#include <ult.h>
int32_t sceUltUlthreadJoin(
        SceUltUlthread *ulthread,
        int32_t *status
)
int32_t sceUltUlthreadTryJoin(
        SceUltUlthread *ulthread,
        int32_t *status
)
```

Arguments

ulthread	User level thread structure
status	Exit code of thread

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*ulthread* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*ulthread* is not 8-byte aligned
`SCE_ULT_ERROR_PERMISSION`	0x80810005	The thread that executed the function cannot go into the wait state (`sceUltUlthreadJoin()` only)
`SCE_ULT_ERROR_STATE`	0x80810006	The termination wait has already completed
`SCE_ULT_ERROR_BUSY`	0x80810007	Other thread is already waiting for the termination
`SCE_ULT_ERROR_AGAIN`	0x80810008	The thread has not yet terminated (`sceUltUlthreadTryJoin()` only)

Description

This function waits for the termination of a user level thread.

If *ulthread* has not yet terminated through `sceUltUlthreadExit()`, `sceUltUlthreadJoin()` waits until *ulthread* terminates, and `sceUltUlthreadTryJoin()` immediately returns an error.

If a value other than NULL is specified for *status*, the exit code of the thread is written to *status*. If *status* is NULL, nothing is written to *status*.

SceUltUlthreadInfo

User level thread information structure

Definition

```
#include <ult.h>
typedef struct SceUltUlthreadInfo{
        uint32_t size;
        uint32_t ulthreadId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t attribute;
        uint32_t ulthreadRuntimeId;
        SceFiberInfo fiberInfo;
        uint64_t __reserved__[(256 - sizeof(uint32_t)*2
                              - (SCE_ULT_MAX_NAME_LENGTH+1)
                              - sizeof(uint32_t)*2
                              - sizeof(SceFiberInfo)
                              / sizeof(uint64_t)];
} SceUltUlthreadInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltUlthreadInfo)`)
ulthreadId	User level thread ID
name	Name of user level thread
attribute	Attribute of user level thread
ulthreadRuntimeId	User level thread runtime ID
fiberInfo	Fiber information used with the user level thread
__reserved__	Reserved area

Description

This structure stores the user level thread information by `sceUltGetUlthreadInfo()`.

For *size*, always first assign `sizeof(SceUltUlthreadInfo)` and then specify it for the function argument.

For `SceFiberInfo`, refer to the "libfiber Reference" document.

sceUltGetUlthreadInfo

Get user level thread status

Definition

```
#include <ult.h>
int32_t sceUltGetUlthreadInfo(
        const SceUltUlthread *ulthread,
        SceUltUlthreadInfo *pInfo
)
```

Arguments

ulthread	Pointer to the user level thread structure
pInfo	Pointer to the user level thread information structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative value) or returns an error code from `sceFiberGetInfo()` for errors. For errors from `sceFiberGetInfo()`, refer to the "libfiber Reference" document.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*ulthread* or *pInfo* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*ulthread* is not 8-byte aligned
SCE_ULT_ERROR_INVALID	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains a user level thread status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

<Sync Library> Waiting Queue Resource Pool

SceUltWaitingQueueResourcePool

Waiting queue resource pool structure

Definition

```
#include <ult.h>
typedef struct SceUltWaitingQueueResourcePool{
        /* private */
} SceUltWaitingQueueResourcePool;
```

Description

This is the waiting queue resource pool structure.

SceUltWaitingQueueResourcePoolOptParam

Waiting queue resource pool options

Definition

```
#include <ult.h>
typedef struct SceUltWaitingQueueResourcePoolOptParam{
        /* private */
} SceUltWaitingQueueResourcePoolOptParam;
```

Description

This structure is used to specify the options during waiting queue resource pool creation.

After executing `sceUltWaitingQueueResourcePoolOptParamInitialize()` to initialize the structure, set values to the member variables for use.

sceUltWaitingQueueResourcePoolOptParamInitialize

Initialize waiting queue resource pool options

Definition

```
#include <ult.h>
int32_t sceUltWaitingQueueResourcePoolOptParamInitialize(
        SceUltWaitingQueueResourcePoolOptParam *optParam
)
```

Arguments

optParam	Option structure of the waiting queue resource pool to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*optParam* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values to all the member variables.

sceUltWaitingQueueResourcePoolGetWorkAreaSize

Get size of work area to be used for waiting queue resource pool

Definition

```
#include <ult.h>
int32_t sceUltWaitingQueueResourcePoolGetWorkAreaSize(
        uint32_t numThreads,
        uint32_t numSyncObjects
)
```

Arguments

numThreads	Maximum number of threads simultaneously access to synchronization objects which are connected to the waiting queue resource pool
numSyncObjects	Number of synchronization objects which are connected to the waiting queue resource pool

Return Values

Size of the work area to be used for the waiting queue resource pool

Description

This function gets the size of the work area required for waiting queue resource pool creation.

Prepare a buffer of the size obtained with this function and specify it to the *workArea* argument of `sceUltWaitingQueueResourcePoolCreate()`.

For details about the arguments, refer to `sceUltWaitingQueueResourcePoolCreate()`.

sceUltWaitingQueueResourcePoolCreate

Create waiting queue resource pool

Definition

```
#include <ult.h>
int32_t sceUltWaitingQueueResourcePoolCreate(
        SceUltWaitingQueueResourcePool *pool,
        const char* name,
        uint32_t numThreads,
        uint32_t numSyncObjects,
        void* workArea,
        SceUltWaitingQueueResourcePoolOptParam *optParam
)
```

Arguments

pool	Pointer to the `SceUltWaitingQueueResourcePool` structure to be initialized
name	Name of the waiting queue resource pool (for debugging). Only the first 31 characters are valid
numThreads	Maximum number of threads simultaneously access to synchronization objects which are connected to a waiting queue resource pool
numSyncObjects	Number of synchronization objects which are connected to a waiting queue resource pool
workArea	Work area of the waiting queue resource pool
optParam	Option of the waiting queue resource pool. If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*pool*, *name*, or *workArea* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*pool*, *workArea*, or *optParam* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	One of the following: - *numThreads* or *numSyncObjects* is 0 - *optParam* is other than NULL and the value is invalid

Description

This function creates a waiting queue resource pool.

A waiting queue resource pool manages a memory area which is temporarily used when a thread for synchronization objects enters a wait state. Resources are allocated from a waiting queue resource pool when a thread entering a wait state, and the resources are freed after the tread becomes executable again.

Specify the number of synchronization objects connected to this waiting queue resource pool to *numSyncObjects*. The synchronization objects are the following: queue data resource pools, queues, semaphores, mutexes, condition variables, and reader/writer locks.

Specify the maximum number of threads simultaneously access to the synchronization objects for *numThreads*.

Specify a buffer of the size obtained with `sceUltWaitingQueueResourcePoolGetWorkAreaSize()` to *`workArea`*. Do not free this buffer until the queues are destroyed by `sceUltWaitingQueueResourcePoolDestroy()`.

sceUltWaitingQueueResourcePoolDestroy

Destroy waiting queue resource pool

Definition

```
#include <ult.h>
int32_t sceUltWaitingQueueResourcePoolDestroy(
        SceUltWaitingQueueResourcePool *pool
)
```

Arguments

pool	Pointer to `SceUltWaitingQueueResourcePool` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*pool* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*pool* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	A thread exists in the waiting queue

Description

This function destroys the waiting queue resource pool.

SceUltWaitingQueueResourcePoolInfo

Waiting queue resource pool information structure

Definition

```
#include <ult.h>
typedef struct SceUltWaitingQueueResourcePoolInfo{
        uint32_t size;
        uint32_t waitingQueueResourcePoolId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t numThreads;
        uint32_t maxNumThreads;
        uint32_t numSyncObjects;
        uint32_t maxNumSyncObjects;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                             - (SCE_ULT_MAX_NAME_LENGTH+1)
                             - sizeof(uint32_t)*4)
                             / sizeof(uint64_t)];
} SceUltWaitingQueueResourcePoolInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltWaitingQueueResourcePoolInfo)`)
waitingQueueResourcePoolId	Waiting queue resource pool ID
name	Name of waiting queue resource pool
numThreads	Number of threads pooled in the waiting queue resource pool
maxNumThreads	Maximum number of threads that can access at the same time to a synchronization object connected to the waiting queue resource pool
numSyncObjects	Number of synchronization objects pooled in the waiting queue resource pool
maxNumSyncObjects	Maximum number of synchronization objects to connect to the waiting queue resource pool
__reserved__	Reserved area

Description

This structure is used for storing waiting queue resource pool information with `sceUltGetWaitingQueueResourcePoolInfo()`.

For *size*, always first assign `sizeof(SceUltWaitingQueueResourcePoolInfo)` and then specify it for the function argument.

sceUltGetWaitingQueueResourcePoolInfo

Get waiting queue resource pool status

Definition

```
#include <ult.h>
int32_t sceUltGetWaitingQueueResourcePoolInfo(
        const SceUltWaitingQueueResourcePool *pool,
        SceUltWaitingQueueResourcePoolInfo *pInfo
)
```

Arguments

pool	Pointer to the waiting queue resource pool structure
pInfo	Pointer to the waiting queue resource pool information structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*pool* or *pInfo* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*pool* or *pInfo* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains the waiting queue resource pool status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

<Sync Library> Queue Data Resource Pool

SceUltQueueDataResourcePool

Queue data resource pool structure

Definition

```
#include <ult.h>
typedef struct SceUltQueueDataResourcePool{
        /* private */
} SceUltQueueDataResourcePool;
```

Description

This is the queue data resource pool structure.

SceUltQueueDataResourcePoolOptParam

Queue data resource pool options

Definition

```
#include <ult.h>
typedef struct SceUltQueueDataResourcePoolOptParam{
        /* private */
} SceUltQueueDataResourcePoolOptParam;
```

Description

This structure is used to specify the options during queue data resource pool creation.

After executing `sceUltQueueDataResourcePoolOptParamInitialize()` to initialize the structure, set values to the member variables for use.

sceUltQueueDataResourcePoolOptParamInitialize

Initialize queue data resource pool options

Definition

```
#include <ult.h>
int32_t sceUltQueueDataResourcePoolOptParamInitialize(
        SceUltQueueDataResourcePoolOptParam *optParam
)
```

Arguments

optParam	Option structure of the queue data resource pool to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*optParam* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values to all the member variables.

sceUltQueueDataResourcePoolGetWorkAreaSize

Get size of work area to be used for queue data resource pool

Definition

```
#include <ult.h>
int32_t sceUltQueueDataResourcePoolGetWorkAreaSize(
        uint32_t numData,
        uint32_t dataSize,
        uint32_t numQueueObjects
)
```

Arguments

numData	Number of buffers used for data in a queue data resource pool
dataSize	Size per buffer for data
numQueueObjects	Number of queue objects that use this queue data resource pool

Return Values

Size of the work area to be used for the queue data resource pool

Description

This function gets the size of the work area required for queue data resource pool creation.

Prepare a buffer of the size obtained with this function and specify it to the *workArea* argument of `sceUltQueueDataResourcePoolCreate()`.

For details about the arguments, refer to `sceUltQueueDataResourcePoolCreate()`.

sceUltQueueDataResourcePoolCreate

Create queue data resource pool

Definition

```
#include <ult.h>
int32_t sceUltQueueDataResourcePoolCreate(
        SceUltQueueDataResourcePool *pool,
        const char* name,
        uint32_t numData,
        uint32_t dataSize,
        uint32_t numQueueObjects,
        SceUltWaitingQueueResourcePool *waitingQueueResourcePool,
        void* workArea,
        SceUltQueueDataResourcePoolOptParam *optParam
)
```

Arguments

pool	Pointer to the `SceUltQueueDataResourcePool` structure to be initialized
name	Name of the queue data resource pool (for debugging). Only the first 31 characters are valid
numData	Number of buffers used for data in a queue data resource pool
dataSize	Size per buffer for data
numQueueObjects	Number of queue objects that use this queue data resource pool
workArea	Work area of queue data resource pool
waitingQueueResourcePool	Waiting queue resource pool for allocating a memory area for a waiting queue
optParam	Option of the queue data resource pool. If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*pool*, *name*, or *workArea* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*pool*, *workArea*, or *optParam* is not 8-byte aligned
SCE_ULT_ERROR_INVALID	0x80810004	One of the following: - *numData*, *dataSize*, or *numQueueObjects* is 0 - *optParam* is other than NULL and the value is invalid
SCE_ULT_ERROR_BUSY	0x80810007	Can no longer connect to *waitingQueueResourcePool*

Description

This function creates queue data resource pool.

A queue data resource pool manages a memory area used when data is pushed to a queue object. After the data being pushed to the queue, a memory area for one piece of data is reserved from the queue data resource pool, and then the data is popped from the queue, the memory area is returned to the queue data resource pool.

Specify `waitingQueueResourcePool` when a memory area for the queue data cannot be allocated and a thread needs to be stopped until the memory area becomes available for the allocation.

Specify the number of buffers used for data in a queue data resource pool to `numData`.

Specify the maximum value of the data size used by queues connected to this queue data resource pool to `dataSize`, and specify the number of the connected queues to `numQueueObjects`.

Specify a buffer of the size obtained with `sceUltQueueDataResourcePoolGetWorkAreaSize()` to `workArea`. Do not free this buffer until the queues are destroyed by `sceUltQueueDataResourcePoolDestroy()`.

sceUltQueueDataResourcePoolDestroy

Destroy queue data resource pool

Definition

```
#include <ult.h>
int32_t sceUltQueueDataResourcePoolDestroy(
        SceUltQueueDataResourcePool *pool
)
```

Arguments

pool	Pointer to `SceUltQueueDataResourcePool` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns the following error code (negative value) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*pool* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*pool* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	A thread exists in the waiting queue

Description

This function destroys the queue data resource pool.

SceUltQueueDataResourcePoolInfo

Queue data resource pool information structure

Definition

```
#include <ult.h>
typedef struct SceUltQueueDataResourcePoolInfo{
        uint32_t size;
        uint32_t queueDataResourcePoolId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t waitingQueueResourcePoolId;
        uint32_t dataSize;
        uint32_t numData;
        uint32_t maxNumData;
        uint32_t numQueueObjects;
        uint32_t maxNumQueueObjects;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                             - (SCE_ULT_MAX_NAME_LENGTH+1)
                             - sizeof(uint32_t)*6)
                             / sizeof(uint64_t)];
} SceUltQueueDataResourcePoolInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltQueueDataResourcePoolInfo)`)
queueDataResourcePoolId	Queue data resource pool ID
name	Name of queue data resource pool
waitingQueueResourcePoolId	ID of waiting queue resource pool for allocating waiting queue memory area
dataSize	Size of one data buffer
numData	Number of data buffers pooled in the queue data resource pool
maxNumData	Maximum number of data buffers in the queue data resource pool
numQueueObjects	Number of queue objects pooled in the queue data resource pool
maxNumQueueObjects	Maximum number of queue objects to be used with the queue data resource pool
__reserved__	Reserved area

Description

This structure is used for storing queue data resource pool information with `sceUltGetQueueDataResourcePoolInfo()`.

For *size*, always first assign `sizeof(SceUltQueueDataResourcePoolInfo)` and then specify it for the function argument.

sceUltGetQueueDataResourcePoolInfo

Get queue data resource pool status

Definition

```
#include <ult.h>
int32_t sceUltGetQueueDataResourcePoolInfo(
        const SceUltQueueDataResourcePool *pool,
        SceUltQueueDataResourcePoolInfo *pInfo
)
```

Arguments

pool	Pointer to the queue data resource pool structure
pInfo	Pointer to the queue data resource pool information structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*pool* or *pInfo* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*pool* or *pInfo* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains the queue data resource pool status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

<Sync Library> Queue

SceUltQueue

Queue structure

Definition

```
#include <ult.h>
typedef struct SceUltQueue{
        /* private */
} SceUltQueue;
```

Description

This is the queue structure.

SceUltQueueOptParam

Queue options

Definition

```
#include <ult.h>
typedef struct SceUltQueueOptParam{
        /* private */
} SceUltQueueOptParam;
```

Description

This structure is used to specify the options during queue creation.

After executing `sceUltQueueOptParamInitialize()` to initialize the structure, set values to the member variables for use.

sceUltQueueOptParamInitialize

Initialize queue options

Definition

```
#include <ult.h>
int32_t sceUltQueueOptParamInitialize(
        SceUltQueueOptParam *optParam
)
```

Arguments

optParam	Option structure of the queue to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*optParam* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values for all the member variables.

sceUltQueueCreate

Create queue

Definition

```
#include <ult.h>
int32_t sceUltQueueCreate(
        SceUltQueue *queue,
        const char* name,
        uint32_t dataSize,
        SceUltQueueDataResourcePool *queueDataResourcePool,
        SceUltWaitingQueueResourcePool *waitingQueueResourcePool,
        SceUltQueueOptParam *optParam
)
```

Arguments

queue	Pointer to the `SceUltQueue` structure to be initialized
name	Name of the queue (for debugging). Only the first 31 characters are valid
dataSize	Size of each data to be input to the queue
queueDataResourcePool	Queue data resource pool used for allocating a buffer for data
waitingQueueResourcePool	Waiting queue resource pool used for allocating a memory area for waiting queue
optParam	Option of the queue. If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*queue*, *name*, or *queueDataResourcePool* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*queue*, *queueDataResourcePool* or *waitingQueueResourcePool* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	One of the following: - *dataSize* is 0, or larger than the maximum data size specified by *queueDataResourcePool* - *optParam* is other than NULL and value is invalid
`SCE_ULT_ERROR_BUSY`	0x80810007	Can no longer connect to `queueDataResourcePool` or `waitingQueueResourcePool`

Description

This function creates a queue.

If NULL is specified for *waitingQueueResourcePool*, `sceUltQueuePush()` and `sceUltQueuePop()` cannot be executed.

sceUltQueuePush, sceUltQueueTryPush

Add data to queue

Definition

```
#include <ult.h>
int32_t sceUltQueuePush(
        SceUltQueue *queue,
        const void *data
)
int32_t sceUltQueueTryPush(
        SceUltQueue *queue,
        const void *data
)
```

Arguments

queue	Pointer to the queue
data	Data to be input to the queue

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*queue* or *data* is NULL
`SCE_ULT_ERROR_PERMISSION`	0x80810005	Execution was done from a thread which cannot enter the wait state (`sceUltQueuePush()` only)
`SCE_ULT_ERROR_STATE`	0x80810006	NULL is specified to `SceUltWaitingQueueResourcePool` pointer of `SceUltQueueDataResourcePool` or cannot transition to the wait state because *queue* has been destroyed (`sceUltQueuePush()` only)
`SCE_ULT_ERROR_BUSY`	0x80810007	The operation could not be executed immediately and a memory area for the waiting queue could not be allocated when entering a wait state was attempted (`sceUltQueuePush()` only)
`SCE_ULT_ERROR_AGAIN`	0x80810008	The memory area for data could not be allocated (`sceUltQueueTryPush()` only)

Description

This function adds data to the *queue*.

Data of the data size specified during creation is copied from *data* to the queue.

If the queue is full when `sceUltQueuePush()` is executed, the thread is in the wait state until there is free capacity in the queue to accept additional data.

sceUltQueuePop, sceUltQueueTryPop

Pop data from queue

Definition

```
#include <ult.h>
int32_t sceUltQueuePop(
        SceUltQueue *queue,
        void *data
)
int32_t sceUltQueueTryPop(
        SceUltQueue *queue,
        void *data
)
```

Arguments

queue	Pointer to the queue
data	Buffer storing the data retrieved from the queue

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*queue* or *data* is NULL
`SCE_ULT_ERROR_PERMISSION`	0x80810005	Execution was done from a thread which cannot enter the wait state (`sceUltQueuePop()` only)
`SCE_ULT_ERROR_STATE`	0x80810006	NULL is specified to *waitingQueueResourcePool* of `sceUltQueueCreate()` (`sceUltQueuePop()` only)
`SCE_ULT_ERROR_BUSY`	0x80810007	The operation could not be executed immediately and a memory area for the waiting queue could not be allocated when entering a wait state was attempted (`sceUltQueuePop()` only)
`SCE_ULT_ERROR_AGAIN`	0x80810008	No data in the queue (`sceUltQueueTryPop()` only)

Description

This function pops data from the *queue*.

Data of the data size specified during creation is copied from the queue to *data*.

If there is no data in the queue when `sceUltQueuePop()` is executed, the thread is in the wait state until data is added to the queue.

sceUltQueueDestroy

Destroy queue

Definition

```
#include <ult.h>
int32_t sceUltQueueDestroy(
        SceUltQueue *queue
)
```

Arguments

queue	Pointer to the `SceUltQueue` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*queue* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*queue* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	A thread exists in the waiting queue

Description

This function destroys the queue.

SceUltWaitThreadInfo

Wait thread information structure

Definition

```
#include <ult.h>
typedef struct SceUltWaitThreadInfo{
        uint32_t size;
        int32_t threadType;
        uint32_t threadId;
        int32_t operation;
        uint64_t __reserved__[(32 - sizeof(uint32_t)*2 - sizeof(int32_t)*2)
                              / sizeof(uint64_t)];
} SceUltWaitThreadInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltWaitThreadInfo)`)
threadType	Thread type
threadId	Thread ID
operation	Operation in synchronization object
__reserved__	Reserved area

Description

This structure is used for storing wait thread information with `sceUltGetMutexInfo()`, `sceUltGetConditionVariableInfo()`, `sceUltGetReaderWriterLockInfo()`, `sceUltGetSemaphoreInfo()`, and `sceUltGetQueueInfo()`.

For *size*, always first assign `sizeof(SceUltWaitThreadInfo)` and then specify it for the function argument.

In *threadType*, the following values are stored.

Value	(Number)	Description
`SCE_ULT_THREAD_TYPE_KERNEL_THREAD`	0x1	Value that indicates a kernel thread
`SCE_ULT_THREAD_TYPE_ULTHREAD`	0x2	Value that indicates a user level thread

In *operation*, 0 is stored for `sceUltGetMutexInfo()` or `sceUltGetConditionVariableInfo()`, one of the following value is stored for `sceUltGetReaderWriterLockInfo()` or `sceUltGetQueueInfo()`, and the number of resources to allocate is stored for `sceUltGetSemaphoreInfo()`.

Value	(Number)	Description
`SCE_ULT_THREAD_OPERATION_READER_LOCK`	0x1	Value that indicates a reader lock
`SCE_ULT_THREAD_OPERATION_WRITER_LOCK`	0x2	Value that indicates a writer lock
`SCE_ULT_THREAD_OPERATION_QUEUE_PUSH`	0x3	Value that indicates a Queue Push
`SCE_ULT_THREAD_OPERATION_QUEUE_POP`	0x4	Value that indicates a Queue Pop

SceUltQueueInfo

Queue information structure

Definition

```
#include <ult.h>
typedef struct SceUltQueueInfo{
        uint32_t size;
        uint32_t queueId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t waitingQueueResourcePoolId;
        uint32_t queueDataResourcePoolId;
        uint32_t dataSize;
        uint32_t numData;
        uint32_t numPushWaitThreads;
        uint32_t numPopWaitThreads;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                             - (SCE_ULT_MAX_NAME_LENGTH+1)
                             - sizeof(uint32_t)*6)
                             / sizeof(uint64_t)];
} SceUltQueueInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltQueueInfo)`)
queueId	Queue ID
name	Name of queue
waitingQueueResourcePoolId	ID of waiting queue resource pool for allocating waiting queue memory area
queueDataResourcePoolId	ID of queue data resource pool for allocating data buffers
dataSize	Size of one datum to insert in the queue
numData	Number of data in the queue
numPushWaitThreads	Number of wait threads for Queue Push
numPopWaitThreads	Number of wait threads for Queue Pop
__reserved__	Reserved area

Description

This structure stores the queue information by `sceUltGetQueueInfo()`.

For *size*, always first assign `sizeof(SceUltQueueInfo)` and then specify it for the function argument.

sceUltGetQueueInfo

Get queue status

Definition

```
#include <ult.h>
int32_t sceUltGetQueueInfo(
        const SceUltQueue *queue,
        SceUltQueueInfo *pInfo,
        SceUltWaitThreadInfo *waitThreadInfoList,
        const uint32_t maxNumThreadInfo
)
```

Arguments

queue	Pointer to the queue structure
pInfo	Pointer to the queue information structure
waitThreadInfoList	Pointer to the wait thread information list or NULL
maxNumThreadInfo	Maximum number of thread information to obtain

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*queue* or *pInfo* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*queue*, *pInfo* or *waitThreadInfoList* is not 8-byte aligned
SCE_ULT_ERROR_INVALID	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains the queue status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

<Sync Library> Semaphore

SceUltSemaphore

Semaphore structure

Definition

```
#include <ult.h>
typedef struct SceUltSemaphore{
        /* private */
} SceUltSemaphore;
```

Description

This is the semaphore structure.

SceUltSemaphoreOptParam

Semaphore options

Definition

```
#include <ult.h>
typedef struct SceUltSemaphoreOptParam{
        /* private */
} SceUltSemaphoreOptParam;
```

Description

This is the structure used to specify the options during semaphore creation.

After executing `sceUltSemaphoreOptParamInitialize()` to initialize the structure, set values to the member variables for use.

sceUltSemaphoreOptParamInitialize

Initialize semaphore options

Definition

```
#include <ult.h>
int32_t sceUltSemaphoreOptParamInitialize(
        SceUltSemaphoreOptParam *optParam
)
```

Arguments

optParam	Option structure of the semaphore to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*optParam* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values for all the member variables.

sceUltSemaphoreCreate

Create semaphore

Definition

```
#include <ult.h>
int32_t sceUltSemaphoreCreate(
        SceUltSemaphore *semaphore,
        const char* name,
        int32_t numInitialResource,
        SceUltWaitingQueueResourcePool *waitingQueueResourcePool,
        SceUltSemaphoreOptParam *optParam
)
```

Arguments

semaphore	Pointer to the `SceUltSemaphore` structure to be initialized
name	Name of the semaphore (for debugging). Only the first 31 characters are valid
numInitialResource	Number of initial resources
waitingQueueResourcePool	Waiting queue resource pool used to allocate a memory area for the waiting queue
optParam	Option of the semaphore. If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*semaphore* or *name* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*semaphore*, *waitingQueueResourcePool* or *optParam* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	*optParam* is other than NULL and the value is invalid
`SCE_ULT_ERROR_BUSY`	0x80810007	Can no longer connect to *waitingQueueResourcePool*

Description

This function creates a semaphore.

A negative value can also be specified for the number of initial resources.

`sceUltSemaphoreAcquire()` cannot be executed if NULL is specified to *waitingQueueResourcePool*.

sceUltSemaphoreAcquire, sceUltSemaphoreTryAcquire

Acquire resources of semaphore

Definition

```
#include <ult.h>
int32_t sceUltSemaphoreAcquire(
        SceUltSemaphore *semaphore,
        uint32_t numResource
)
int32_t sceUltSemaphoreTryAcquire(
        SceUltSemaphore *semaphore,
        uint32_t numResource
)
```

Arguments

semaphore	Pointer to the semaphore
numResource	Number of resources to be acquired

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*semaphore* is NULL
`SCE_ULT_ERROR_RANGE`	0x80810003	*numResource* is 0 or lower, or 0x80000000 or higher
`SCE_ULT_ERROR_PERMISSION`	0x80810005	Execution was done from a thread which cannot enter the wait state (`sceUltSemaphoreAcquire()` only)
`SCE_ULT_ERROR_STATE`	0x80810006	NULL is specified for *waitingQueueResourcePool* in `sceUltSemaphoreCreate()` or cannot transition to the wait state because *semaphore* has been destroyed (`sceUltSemaphoreAcquire()` only)
`SCE_ULT_ERROR_BUSY`	0x80810007	The operation could not be executed immediately and a memory area for the waiting queue could not be allocated when entering a wait state was attempted (`sceUltSemaphoreAcquire()` only)
`SCE_ULT_ERROR_AGAIN`	0x80810008	The requested resource could not be acquired (`sceUltSemaphoreTryAcquire()` only)

Description

This function acquires the resources of the semaphore.

If the resource(s) cannot be acquired immediately when `sceUltSemaphoreAcquire()` is executed, the thread is in the wait state until the resource(s) is acquired.

sceUltSemaphoreRelease

Release resources to semaphore

Definition

```
#include <ult.h>
int32_t sceUltSemaphoreRelease(
        SceUltSemaphore *semaphore,
        uint32_t numResource
)
```

Arguments

semaphore	Pointer to semaphore
numResource	Number of resources to be released

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*semaphore* is NULL
`SCE_ULT_ERROR_RANGE`	0x80810003	*numResource* is 0 or lower, or 0x80000000 or higher
`SCE_ULT_ERROR_STATE`	0x80810006	When this operation is executed, the total number of resources exceeds the upper limit (0x7fffffff)

Description

This function releases the resources of the semaphore.

sceUltSemaphoreDestroy

Destroy semaphore

Definition

```
#include <ult.h>
int32_t sceUltSemaphoreDestroy(
        SceUltSemaphore *semaphore
)
```

Arguments

semaphore	Pointer to `SceUltSemaphore` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*semaphore* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*semaphore* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	A thread exists in the waiting queue

Description

This function destroys the semaphore.

SceUltSemaphoreInfo

Semaphore information structure

Definition

```
#include <ult.h>
typedef struct SceUltSemaphoreInfo{
        uint32_t size;
        uint32_t semaphoreId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t waitingQueueResourcePoolId;
        uint32_t numCurrentResource;
        uint32_t numWaitThreads;
        uint32_t reserved0;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                             - (SCE_ULT_MAX_NAME_LENGTH+1)
                             - sizeof(uint32_t)*4)
                             / sizeof(uint64_t)];
} SceUltSemaphoreInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltSemaphoreInfo)`)
semaphoreId	Semaphore ID
name	Name of semaphore
waitingQueueResourcePoolId	ID of waiting queue resource pool for allocating waiting queue memory area
numCurrentResource	Current number of resources
numWaitThreads	Number of wait threads
reserved0	Reserved area
__reserved__	Reserved area

Description

This structure stores the semaphore information by `sceUltGetSemaphoreInfo()`.

For *size*, always first assign `sizeof(SceUltSemaphoreInfo)` and then specify it for the function argument.

sceUltGetSemaphoreInfo

Get semaphore status

Definition

```
#include <ult.h>
int32_t sceUltGetSemaphoreInfo (
        const SceUltSemaphore *semaphore,
        SceUltSemaphoreInfo *pInfo,
        SceUltWaitThreadInfo *waitThreadInfoList,
        const uint32_t maxNumThreadInfo
)
```

Arguments

semaphore	Pointer to the semaphore structure
pInfo	Pointer to the semaphore information structure
waitThreadInfoList	Pointer to the wait thread information list or NULL
maxNumThreadInfo	Maximum number of thread information to obtain

Return Values

Returns SCE_OK(0) for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*semaphore* or *pInfo* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*semaphore*, *pInfo* or *waitThreadInfoList* is not 8-byte aligned
SCE_ULT_ERROR_INVALID	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains the semaphore status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

<Sync Library> Mutex

SceUltMutex

Mutex structure

Definition

```
#include <ult.h>
typedef struct SceUltMutex{
        /* private */
} **SceUltMutex;**
```

Description

This is the mutex structure.

SceUltMutexOptParam

Mutex options

Definition

```
#include <ult.h>
typedef struct SceUltMutexOptParam{
        uint32_t attribute
        /* private */
} SceUltMutexOptParam;
```

Members

attribute	Attribute of mutex (Default value: 0)

Description

This is the structure used to specify the options to be used during mutex creation.

After executing `sceUltMutexOptParamInitialize()` to initialize the structure, set values to the member variables for use.

Specify the bitwise OR of the following value to *attribute*. The default value is 0.

Value	(Number)	Description
SCE_ULT_MUTEX_ATTRIBUTE_RECURSIVE	0x1	Enables mutex recursive locking for mutexes

sceUltMutexOptParamInitialize

Initialize mutex options

Definition

```
#include <ult.h>
int32_t sceUltMutexOptParamInitialize(
        SceUltMutexOptParam *optParam
)
```

Arguments

optParam	Option structure of the mutex to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*optParam* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values for all the member variables.

sceUltMutexCreate

Create mutex

Definition

```
#include <ult.h>
int32_t sceUltMutexCreate(
        SceUltMutex *mutex,
        const char* name,
        SceUltWaitingQueueResourcePool *waitingQueueResourcePool,
        SceUltMutexOptParam *optParam
)
```

Arguments

mutex	Pointer to the `SceUltMutex` structure to be initialized
name	Name of the mutex (for debugging). Only the first 31 characters are valid
waitingQueueResourcePool	Waiting queue resource pool used to allocate a memory area for the waiting queue
optParam	Mutex option If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*mutex*, *name* or *waitingQueueResourcePool* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*mutex*, *waitingQueueResourcePool*, or *optParam* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	*optParam* is other than NULL and the value is invalid
`SCE_ULT_ERROR_BUSY`	0x80810007	Can no longer connect to *waitingQueueResourcePool*

Description

This function creates a mutex.

sceUltMutexLock, sceUltMutexTryLock

Lock mutex

Definition

```
#include <ult.h>
int32_t sceUltMutexLock(
        SceUltMutex *mutex
)
int32_t sceUltMutexTryLock(
        SceUltMutex *mutex
)
```

Arguments

mutex	Pointer to the mutex

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*mutex* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*mutex* is not 8-byte aligned
`SCE_ULT_ERROR_PERMISSION`	0x80810005	Execution was done from a thread which cannot enter the wait state (`sceUltMutexLock()` only)
`SCE_ULT_ERROR_BUSY`	0x80810007	A memory area for the waiting queue could not be allocated
`SCE_ULT_ERROR_STATE`	0x80810006	A thread that has acquired a lock attempted to lock a mutex for which recursive locking is not allowed again
`SCE_ULT_ERROR_AGAIN`	0x80810008	The mutex is already locked by another thread (`sceUltMutexTryLock()` only)

Description

This function locks mutexes.

When locking is successful, the thread that executed this function becomes the owner of the mutex. The information of the owner is separated between the thread of the OS and the user level thread.

Only the owner of the mutex can unlock the mutex.

If locking cannot be achieved immediately when `sceUltMutexLock()` is executed, the thread is in the wait state until locking is achieved.

sceUltMutexUnlock

Unlock mutex

Definition

```
#include <ult.h>
int32_t sceUltMutexUnlock(
        SceUltMutex *mutex
)
```

Arguments

mutex	Pointer to the mutex

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*mutex* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*mutex* is not 8-byte aligned
`SCE_ULT_ERROR_PERMISSION`	0x80810005	This thread is not the thread for which the owner of the *mutex* executed this function

Description

This function unlocks the mutex.

sceUltMutexDestroy

Destroy mutex

Definition

```
#include <ult.h>
int32_t sceUltMutexDestroy(
        SceUltMutex *mutex
)
```

Arguments

mutex Pointer to the `SceUltMutex` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*mutex* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*mutex* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	One of the following: - This mutex is locked - A thread exists in the waiting queue - Condition variables that belong to this mutex exist

Description

This function destroys the mutex.

SceUltMutexInfo

Mutex information structure

Definition

```
#include <ult.h>
typedef struct SceUltMutexInfo{
        uint32_t size;
        uint32_t mutexId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t attribute;
        uint32_t waitingQueueResourcePoolId;
        uint32_t currentOwnerId;
        uint32_t recursiveLockCount;
        uint32_t numWaitThreads;
        int32_t currentOwnerThreadType;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                           - (SCE_ULT_MAX_NAME_LENGTH+1)
                           - sizeof(uint32_t)*5
                           - sizeof(int32_t))
                           / sizeof(uint64_t)];
} SceUltMutexInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltMutexInfo)`)
mutexId	Mutex ID
name	Name of mutex
attribute	Attribute of mutex
waitingQueueResourcePoolId	ID of waiting queue resource pool for allocating the wait queue memory area
currentOwnerId	Currently locked thread ID
recursiveLockCount	Recursive lock count of mutex
numWaitThreads	Number of wait threads
currentOwnerThreadType	Type of thread that is currently locked
__reserved__	Reserved area

Description

This structure stores the mutex information by `sceUltGetMutexInfo()`.

For *size*, always first assign `sizeof(SceUltMutexInfo)` and then specify it for the function argument.

When `SCE_ULT_MUTEX_ATTRIBUTE_RECURSIVE` is not specified for *attribute*, the *recursiveLockCount* value is undefined.

When *currentOwnerId* is 0, it means that there are no locked threads.

When *currentOwnerId* is not 0, one of the following values will be stored in *currentOwnerThreadType*.

Value	(Number)	Description
`SCE_ULT_THREAD_TYPE_KERNEL_THREAD`	0x1	Value indicating a kernel thread
`SCE_ULT_THREAD_TYPE_ULTHREAD`	0x2	Value indicating a user level thread

sceUltGetMutexInfo

Get mutex status

Definition

```
#include <ult.h>
int32_t sceUltGetMutexInfo (
        const SceUltMutex *mutex,
        SceUltMutexInfo *pInfo,
        SceUltWaitThreadInfo *waitThreadInfoList,
        const uint32_t maxNumThreadInfo
)
```

Arguments

mutex	Pointer to the mutex structure
pInfo	Pointer to the mutex information structure
waitThreadInfoList	Pointer to the wait thread information list or NULL
maxNumThreadInfo	Maximum number of thread information to obtain

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*mutex* or *pInfo* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*mutex*, *pInfo* or *waitThreadInfoList* is not 8-byte aligned
SCE_ULT_ERROR_INVALID	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains the mutex status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

<Sync Library> Condition Variables

SceUltConditionVariable

Condition variable structure

Definition

```
#include <ult.h>
typedef struct SceUltConditionVariable{
        /* private */
} SceUltConditionVariable;
```

Description

This is the condition variable structure.

SceUltConditionVariableOptParam

Condition variable options

Definition

```
#include <ult.h>
typedef struct SceUltConditionVariableOptParam{
        /* private */
} SceUltConditionVariableOptParam;
```

Description

This is the structure used to specify the options to be used during condition variable creation.

After executing `sceUltConditionVariableOptParamInitialize()` to initialize the structure, set values to the member variables for use.

sceUltConditionVariableOptParamInitialize

Initialize condition variable options

Definition

```
#include <ult.h>
int32_t sceUltConditionVariableOptParamInitialize(
        SceUltConditionVariableOptParam *optParam
)
```

Arguments

optParam	Option structure of the condition variables to be initialized

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*optParam* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values for all the member variables.

sceUltConditionVariableCreate

Create condition variable

Definition

```
#include <ult.h>
int32_t sceUltConditionVariableCreate(
        SceUltConditionVariable *cv,
        const char* name,
        SceUltMutex *mutex,
        SceUltConditionVariableOptParam *optParam
)
```

Arguments

cv	Pointer to the condition variable structure to be initialized
name	Name of the condition variable (for debugging). Only the first 31 characters are valid
mutex	Mutex to be used
optParam	Option of the condition variable. If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	cv, *mutex* or *name* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	cv, *mutex* or *optParam* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	*optParam* is other than NULL and the value is invalid
`SCE_ULT_ERROR_PERMISSION`	0x80810005	`SCE_ULT_MUTEX_ATTRIBUTE_RECURSIVE` is specified to *mutex*
`SCE_ULT_ERROR_BUSY`	0x80810007	Can no longer connect to the waiting queue resource pool connected to *mutex*

Description

This function creates a condition variable.

A waiting queue resource pool connected to *mutex* is used.

sceUltConditionVariableSignal, sceUltConditionVariableSignalAll

Send signal to condition variables

Definition

```
#include <ult.h>
int32_t sceUltConditionVariableSignal(
        SceUltConditionVariable *cv
)
int32_t sceUltConditionVariableSignalAll(
        SceUltConditionVariable *cv
)
```

Arguments

cv	Pointer to the condition variable

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*cv* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*cv* is not 8-byte aligned

Description

This function sends a signal to the condition variable.

If there are threads waiting for signals for condition variables, `sceUltConditionVariableSignal()` resumes execution of one of these threads, and `sceUltConditionVariableSignalAll()` resumes execution of all the threads.

If there are no threads waiting for signals, nothing is done.

sceUltConditionVariableWait

Wait for signal at condition variable

Definition

```
#include <ult.h>
int32_t sceUltConditionVariableWait(
        SceUltConditionVariable *cv
)
```

Arguments

`cv` Pointer to the condition variable

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*cv* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*cv* is not 8-byte aligned
SCE_ULT_ERROR_PERMISSION	0x80810005	The owner of the mutex using *cv* is not the thread that executed this function, or the thread that executed the function cannot enter the wait state
SCE_ULT_ERROR_STATE	0x80810006	Cannot transition to the wait state because *cv* has been destroyed

Description

Threads are put in standby until the condition variable gets a signal.

At the start of standby, the mutex is unlocked, and when execution is resumed, the mutex is locked again.

sceUltConditionVariableDestroy

Destroy condition variable

Definition

```
#include <ult.h>
int32_t sceUltConditionVariableDestroy(
        SceUltConditionVariable *cv
)
```

Arguments

cv Pointer to the SceUltConditionVariable structure

Return Values

Returns SCE_OK(0) for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*cv* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*cv* is not 8-byte aligned
SCE_ULT_ERROR_STATE	0x80810006	There is a thread waiting for a signal

Description

This function destroys the condition variable.

SceUltConditionVariableInfo

Condition variable information structure

Definition

```
#include <ult.h>
typedef struct SceUltConditionVariableInfo{
        uint32_t size;
        uint32_t cvId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t mutexId;
        uint32_t numWaitThreads;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                              - (SCE_ULT_MAX_NAME_LENGTH+1)
                              - sizeof(uint32_t)*2)
                              / sizeof(uint64_t)];
} SceUltConditionVariableInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltConditionVariableInfo)`)
cvId	Condition variable ID
name	Name of condition variable
mutexId	Mutex ID being used
numWaitThreads	Number of wait threads
__reserved__	Reserved area

Description

This structure stores the condition variable information by `sceUltGetConditionVariableInfo()`.

For *size*, always first assign `sizeof(SceUltConditionVariableInfo)` and then specify it for the function argument.

sceUltGetConditionVariableInfo

Get condition variable status

Definition

```
#include <ult.h>
int32_t sceUltGetConditionVariableInfo (
        const SceUltConditionVariable *cv,
        SceUltConditionVariableInfo *pInfo,
        SceUltWaitThreadInfo *waitThreadInfoList,
        const uint32_t maxNumThreadInfo
)
```

Arguments

cv	Pointer to the condition variable structure
pInfo	Pointer to the condition variable information structure
waitThreadInfoList	Pointer to the wait thread information list or NULL
maxNumThreadInfo	Maximum number of thread information to obtain

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*cv* or *pInfo* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*cv*, *pInfo* or *waitThreadInfoList* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This thread obtains the condition variable status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

<Sync Library> Reader/Writer Lock

SceUltReaderWriterLock

Reader/writer lock structure

Definition

```
#include <ult.h>
typedef struct SceUltReaderWriterLock{
        /* private */
} SceUltReaderWriterLock;
```

Description

This is the reader/writer lock structure.

SceUltReaderWriterLockOptParam

Reader/writer lock options

Definition

```
#include <ult.h>
typedef struct SceUltReaderWriterLockOptParam{
        /* private */
} SceUltReaderWriterLockOptParam;
```

Description

This is the structure used to specify options during reader/writer lock creation.

After executing `sceUltReaderWriterLockOptParamInitialize()` to initialize the structure, set values to the member variables for use.

sceUltReaderWriterLockOptParamInitialize

Initialize reader/writer lock options

Definition

```
#include <ult.h>
int32_t sceUltReaderWriterLockOptParamInitialize(
        SceUltReaderWriterLockOptParam *optParam
)
```

Arguments

optParam	Option structure of the reader/writer lock

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*optParam* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*optParam* is not 8-byte aligned

Description

This function initializes *optParam* and sets the default values for all the member variables.

sceUltReaderWriterLockCreate

Create reader/writer lock

Definition

```
#include <ult.h>
int32_t sceUltReaderWriterLockCreate(
        SceUltReaderWriterLock *rwlock,
        const char* name,
        SceUltWaitingQueueResourcePool *waitingQueueResourcePool,
        SceUltReaderWriterLockOptParam *optParam
)
```

Arguments

rwlock	Pointer to the `SceUltReaderWriterLock` structure
name	Name of the reader/writer lock (for debugging). Only the first 31 characters are valid
waitingQueueResourcePool	Waiting queue resource pool used to allocate a memory area for the waiting queue
optParam	Option. If NULL, the default option is used

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*rwlock* or *name* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*rwlock*, *waitingQueueResourcePool* or *optParam* is not 8-byte aligned
`SCE_ULT_ERROR_INVALID`	0x80810004	*optParam* is other than NULL and the value is invalid
`SCE_ULT_ERROR_BUSY`	0x80810007	Can no longer connect to *waitingQueueResourcePool*

Description

This function creates reader/writer locks.

sceUltReaderWriterLockLockRead, sceUltReaderWriterLockTryLockRead

Set reader lock

Definition

```
#include <ult.h>
int32_t sceUltReaderWriterLockLockRead(
        SceUltReaderWriterLock *rwlock
)
int32_t sceUltReaderWriterLockTryLockRead(
        SceUltReaderWriterLock *rwlock
)
```

Arguments

rwlock	Pointer to the `SceUltReaderWriterLock` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*rwlock* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*rwlock* is not 8-byte aligned
`SCE_ULT_ERROR_PERMISSION`	0x80810005	Execution was done from a thread which cannot enter the wait state (`sceUltReaderWriterLockLockRead()` only)
`SCE_ULT_ERROR_STATE`	0x80810006	Cannot transition to the wait state because *rwlock* has been destroyed (`sceUltReaderWriterLockLockRead()` only)
`SCE_ULT_ERROR_BUSY`	0x80810007	The operation could not be executed immediately and a memory area for the waiting queue could not be allocated when entering a wait state was attempted (`sceUltReaderWriterLockLockRead()` only)
`SCE_ULT_ERROR_AGAIN`	0x80810008	The lock could not be immediately set, either because the writer lock is set, or because there is a thread waiting for writer lock (`sceUltReaderWriterLockTryLockRead()` only)

Description

This function sets the reader lock.

If a lock cannot be set immediately when `sceUltReaderWriterLockLockRead()` is executed, this function keeps the thread in the wait state until a lock can be set. If multiple threads enter wait state for a reader/writer lock, a thread waiting queue will be created. Threads will be placed in the wait queue in the order in which they have entered waiting state, with the thread that has entered waiting state first at the top.

sceUltReaderWriterLockUnlockRead

Release reader lock

Definition

```
#include <ult.h>
int32_t sceUltReaderWriterLockUnlockRead(
        SceUltReaderWriterLock *rwlock
)
```

Arguments

rwlock	Pointer to the `SceUltReaderWriterLock` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*rwlock* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*rwlock* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	Reader lock is not set

Description

This function releases the reader lock.

Reader lock release can be executed from any thread.

sceUltReaderWriterLockLockWrite, sceUltReaderWriterLockTryLockWrite

Set writer lock

Definition

```
#include <ult.h>
int32_t sceUltReaderWriterLockLockWrite(
        SceUltReaderWriterLock *rwlock
)
int32_t sceUltReaderWriterLockTryLockWrite(
        SceUltReaderWriterLock *rwlock
)
```

Arguments

rwlock	Pointer to the `SceUltReaderWriterLock` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*rwlock* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*rwlock* is not 8-byte aligned
`SCE_ULT_ERROR_PERMISSION`	0x80810005	Execution was done from a thread which cannot enter the wait state (`sceUltReaderWriterLockLockWrite()` only)
`SCE_ULT_ERROR_STATE`	0x80810006	Cannot transition to the wait state because *rwlock* has been destroyed (`sceUltReaderWriterLockLockWrite()` only)
`SCE_ULT_ERROR_BUSY`	0x80810007	The operation could not be executed immediately and a memory area for the waiting queue could not be allocated when entering a wait state was attempted (`sceUltReaderWriterLockLockWrite()` only)
`SCE_ULT_ERROR_AGAIN`	0x80810008	A lock could not be set because either reader lock or writer lock is set (`sceUltReaderWriterLockTryLockWrite()` only)

Description

This function sets the writer lock.

If a lock cannot be set immediately when `sceUltReaderWriterLockLockWrite()` is executed, this function keeps the thread in the wait state until a lock can be set. If multiple threads enter wait state for a reader/writer lock, a thread waiting queue will be created. Threads will be placed in the wait queue in the order in which they have entered waiting state, with the thread that has entered waiting state first at the top.

sceUltReaderWriterLockUnlockWrite

Release writer lock

Definition

```
#include <ult.h>
int32_t sceUltReaderWriterLockUnlockWrite(
        SceUltReaderWriterLock *rwlock
)
```

Arguments

rwlock	Pointer to the `SceUltReaderWriterLock` structure

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
`SCE_ULT_ERROR_NULL`	0x80810001	*rwlock* is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	*rwlock* is not 8-byte aligned
`SCE_ULT_ERROR_STATE`	0x80810006	Writer lock is not set

Description

This function releases the writer lock.

Writer lock release can be executed from any thread.

sceUltReaderWriterLockDestroy

Reader/writer lock termination processing

Definition

```
#include <ult.h>
int32_t sceUltReaderWriterLockDestroy(
        SceUltReaderWriterLock *rwlock
)
```

Arguments

rwlock	Pointer to the SceUltReaderWriterLock structure

Return Values

Returns SCE_OK(0) for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*rwlock* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*rwlock* is not 8-byte aligned
SCE_ULT_ERROR_STATE	0x80810006	A thread exists in the waiting queue

Description

This function terminates the reader/writer lock.

SceUltReaderWriterLockInfo

Reader/writer lock information structure

Definition

```
#include <ult.h>
typedef struct SceUltReaderWriterLockInfo{
        uint32_t size;
        uint32_t rwLockId;
        char name[SCE_ULT_MAX_NAME_LENGTH+1];
        uint32_t waitingQueueResourcePoolId;
        uint32_t lockStatus;
        uint32_t numLockingReaders;
        uint32_t writeOwnerId;
        uint32_t numWaitThreads;
        int32_t writeOwnerThreadType;
        uint64_t __reserved__[(128 - sizeof(uint32_t)*2
                             - (SCE_ULT_MAX_NAME_LENGTH+1)
                             - sizeof(uint32_t)*5
                             - sizeof(int32_t))
                             / sizeof(uint64_t)];
} SceUltReaderWriterLockInfo;
```

Members

size	Size of this structure (value of `sizeof(SceUltReaderWriterLockInfo)`)
rwLockId	Reader/writer lock ID
name	Name of reader/writer lock
waitingQueueResourcePoolId	ID of the waiting queue resource pool for allocating the wait queue memory area
lockStatus	Lock status
numLockingReaders	Number of reader locked threads
writeOwnerId	ID of writer locked thread
numWaitThreads	Number of wait threads
writeOwnerThreadType	Type of thread that is writer locked
__reserved__	Reserved area

Description

This structure stores the reader/writer lock information by `sceUltGetReaderWriterLockInfo()`.

For *size*, always first assign `sizeof(SceUltReaderWriterLockInfo)` and then specify it for the function argument.

In *lockStatus*, the following values are stored.

Value	(Number)	Description
`SCE_ULT_READER_WRITER_LOCK_STATUS_UNLOCK`	0x0	Value that indicates an unlock status
`SCE_ULT_READER_WRITER_LOCK_STATUS_READER_LOCK`	0x1	Value that indicates an reader lock status
`SCE_ULT_READER_WRITER_LOCK_STATUS_WRITER_LOCK`	0x2	Value that indicates an writer lock status

When *lockStatus* is not SCE_ULT_READER_WRITER_LOCK_STATUS_READER_LOCK, the *numLockingReaders* value will be undefined.

When *lockStatus* is not SCE_ULT_READER_WRITER_LOCK_STATUS_WRITER_LOCK, the *writeOwnerId* and *writeOwnerThreadType* values will be undefined.

When *lockStatus* is SCE_ULT_READER_WRITER_LOCK_STATUS_WRITER_LOCK, one of the following values will be stored in *writeOwnerThreadType*.

Value	(Number)	Description
SCE_ULT_THREAD_TYPE_KERNEL_THREAD	0x1	Value indicating a kernel thread
SCE_ULT_THREAD_TYPE_ULTHREAD	0x2	Value indicating a user level thread

sceUltGetReaderWriterLockInfo

Get reader/writer lock status

Definition

```
#include <ult.h>
int32_t sceUltGetReaderWriterLockInfo (
        const SceUltReaderWriterLock *rwlock,
        SceUltReaderWriterLockInfo *pInfo,
        SceUltWaitThreadInfo *waitThreadInfoList,
        const uint32_t maxNumThreadInfo
)
```

Arguments

rwlock	Pointer to the reader/writer lock structure
pInfo	Pointer to the reader/writer lock information structure
waitThreadInfoList	Pointer to the wait thread information list or NULL
maxNumThreadInfo	Maximum number of thread information to obtain

Return Values

Returns `SCE_OK(0)` for normal termination.

Returns one of the following error codes (negative values) in case of an error.

Value	(Number)	Description
SCE_ULT_ERROR_NULL	0x80810001	*rwlock* or *pInfo* is NULL
SCE_ULT_ERROR_ALIGNMENT	0x80810002	*rwlock*, *pInfo* or *waitThreadInfoList* is not 8-byte aligned
SCE_ULT_ERROR_INVALID	0x80810004	The specified argument value is invalid (the value for *size* in the information structure is invalid)

Description

This function obtains the reader/writer lock status.

It is provided for debugging support. Since it does not synchronize, there is a possibility coherent information will not be obtained, and the information that can be obtained changes moment by moment. Do not perform programming that changes the control with the obtained information.

Constants

Return Codes

List of return codes returned by libult

Definition

Value	(Number)	Description
`SCE_OK`	0x00000000	Successful
`SCE_ULT_ERROR_NULL`	0x80810001	The specified pointer is NULL
`SCE_ULT_ERROR_ALIGNMENT`	0x80810002	The alignment restrictions are not met
`SCE_ULT_ERROR_RANGE`	0x80810003	The specified value is out of range
`SCE_ULT_ERROR_INVALID`	0x80810004	The specified argument value is invalid
`SCE_ULT_ERROR_PERMISSION`	0x80810005	This is an unauthorized operation
`SCE_ULT_ERROR_STATE`	0x80810006	The current state does not allow application of this operation
`SCE_ULT_ERROR_BUSY`	0x80810007	The resource cannot be used
`SCE_ULT_ERROR_AGAIN`	0x80810008	Execution of this operation is temporarily not possible
`SCE_ULT_ERROR_FATAL`	0x80810009	A fatal error occurred