libcodecengine Reference

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Document serial number: 000004892117

Table of Contents

Datatypes

SceCodecEnginePmonProcessorLoad

Codec Engine processor load structure

Definition

```
#include <codecengine.h>
typedef struct SceCodecEnginePmonProcessorLoad {
        SceUInt32 size;
        SceUInt32 average;
} SceCodecEnginePmonProcessorLoad;
```

Members

size	Size of the structure
average	Relative load to all the processing resources on Codec Engine (%)

Description

This structure represents the Codec Engine processor load.

By passing this structure to the libcodecengine API, the Codec Engine processor load can be obtained. To *size*, specify `sizeof(SceCodecEnginePmonProcessorLoad)`.

See Also

`sceCodecEnginePmonStart()`, `sceCodecEnginePmonStop()`, `sceCodecEnginePmonGetProcessorLoad()`, `sceCodecEnginePmonReset()`

APIs

sceCodecEngineOpenUnmapMemBlock

Remap as memory that is enabled for reading and writing by the Codec Engine but not by the user

Definition

```
#include <codecengine.h>
SceUID sceCodecEngineOpenUnmapMemBlock (
        void *pMemBlock,
        SceUInt32 memBlockSize
)
```

Arguments

pMemBlock	Starting address of memory enabled for reading and writing by the user
memBlockSize	Size of memory enabled for reading and writing by the user

Return Values

Value	Description
>0	Memory block ID (UID)
<0	Error

Description

This function unmaps the cache-disabled and physical continuous memory that is enabled for reading and writing by the user, and then remaps as a cache-disabled and physical continuous memory that is enabled for reading and writing by the Codec Engine but not by the user. This API returns an identifier to the remapped memory area. This API can be applied at the same time to up to four memory areas.

Specify the memory areas allocated with `sceKernelAllocMemBlock()` to this API. When allocating on LPDDR2, specify `SCE_KERNEL_MEMBLOCK_TYPE_USER_MAIN_PHYCONT_NC_RW` to the argument, and when allocating on the CDRAM, specify `SCE_KERNEL_MEMBLOCK_TYPE_USER_CDRAM_RW` to the argument. Refer to the "Notes" chapter in the "libcodecengine Overview" document for additional details.

Notes

This function is multi-thread safe.

See Also

`sceCodecEngineCloseUnmapMemBlock()`,
`sceCodecEngineAllocMemoryFromUnmapMemBlock()`

sceCodecEngineCloseUnmapMemBlock

Remap as a memory that is enabled for reading and writing by the user

Definition

```
#include <codecengine.h>
SceInt32 sceCodecEngineCloseUnmapMemBlock (
        SceUID uid
)
```

Arguments

uid Memory block ID

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error

Description

This function unmaps the cache-disabled and physical continuous memory that is enabled for reading and writing by the Codec Engine but not by the user, and then remaps as a cache-disabled and physical continuous memory that is enabled for reading and writing by the user. Before executing this API, all memory allocated with `sceCodecEngineAllocMemoryFromUnmapMemBlock()` must be released with `sceCodecEngineFreeMemoryFromUnmapMemBlock()`.

Notes

This function is multi-thread safe.

See Also

`sceCodecEngineOpenUnmapMemBlock()`,
`sceCodecEngineFreeMemoryFromUnmapMemBlock()`

sceCodecEngineAllocMemoryFromUnmapMemBlock

Allocate memory from a memory area that is enabled for reading and writing by the Codec Engine but not by the user

Definition

```
#include <codecengine.h>
SceUIntVAddr sceCodecEngineAllocMemoryFromUnmapMemBlock (
        SceUID uid,
        SceUInt32 size,
        SceUInt32 alignment
)
```

Arguments

uid	Memory block ID
size	Requested memory size
alignment	Alignment size of Requested memory

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error

Description

This function allocates memory from a cache-disabled and physical continuous memory area that is enabled for reading and writing by the Codec Engine but not by the user. The requested memory size and alignment size can be specified.

For points to consider when using memory allocated with this API, refer to "Notes" in the "libcodecengine Overview" document.

Notes

This function is multi-thread safe.

See Also

sceCodecEngineFreeMemoryFromUnmapMemBlock(),
sceCodecEngineOpenUnmapMemBlock()

sceCodecEngineFreeMemoryFromUnmapMemBlock

Release memory to a memory area that is enabled for reading and writing by the Codec Engine but not by the user

Definition

```
#include <codecengine.h>
SceInt32 sceCodecEngineFreeMemoryFromUnmapMemBlock (
        SceUID uid,
        SceUIntVAddr p
)
```

Arguments

uid Memory block ID
p Memory allocated with `sceCodecEngineAllocMemoryFromUnmapMemBlock()`

Return Values

Value	Description
0 (`SCE_OK`)	Success
<0	Error

Description

This function releases memory to a cache-disabled and physical continuous memory area that is enabled for reading and writing by the Codec Engine but not by the user.

Notes

This function is multi-thread safe.

See Also

`sceCodecEngineAllocMemoryFromUnmapMemBlock()`,
`sceCodecEngineCloseUnmapMemBlock()`

Debug APIs

sceCodecEnginePmonStart

Start measuring Codec Engine performance

Definition

```
#include <codecengine.h>
SceInt32 sceCodecEnginePmonStart (
        void
)
```

Arguments

None

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error

Description

Starts measuring Codec Engine performance.

The measurement segment is defined between when this function and the `sceCodecEnginePmonStop()` function are called.

Notes

This function is not multi-thread safe.

See Also

`SceCodecEnginePmonProcessorLoad`, `sceCodecEnginePmonStop()`

sceCodecEnginePmonStop

Stop measuring Codec Engine performance

Definition

```
#include <codecengine.h>
SceInt32 sceCodecEnginePmonStop (
        void
)
```

Arguments

None

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error

Description

Stops measuring Codec Engine performance.

The measurement segment is defined between when the `sceCodecEnginePmonStart()` function and this function are called.

Notes

This function is not multi-thread safe.

See Also

`SceCodecEnginePmonProcessorLoad`, `sceCodecEnginePmonStart()`

sceCodecEnginePmonGetProcessorLoad

Obtain Codec Engine processor load

Definition

```
#include <codecengine.h>
SceInt32 sceCodecEnginePmonGetProcessorLoad (
        SceCodecEnginePmonProcessorLoad *pProcessorLoad
)
```

Arguments

pProcessorLoad Pointer to `SceCodecEnginePmonProcessorLoad` structure

Return Values

Value	Description
0 (`SCE_OK`)	Success
<0	Error

Description

Obtains the Codec Engine processor load.

Before executing this function, the measurement must be executed by using `sceCodecEnginePmonStart()` and `sceCodecEnginePmonStop()`.Unless `sceCodecEnginePmonReset()` is executed, measurement results will be accumulated. Note that depending on the upper limit value of the counter being used for performance measurement, a correct value may not be obtained when measuring processor load over a long period of time. Because of this, it is recommended that the measurement segment be approximately one granularity of audio processing (such as 256 samples) or within one second at the longest.

Notes

This function is not multi-thread safe.

Examples

```
SceCodecEnginePmonProcessorLoad ProcessorLoad;

// Start measuring performance
sceCodecEnginePmonStart();

// do something

// Stop measuring performance
sceCodecEnginePmonStop();

// Obtain Codec Engine processor load
ProcessorLoad.size = sizeof(ProcessorLoad);
sceCodecEnginePmonGetProcessorLoad(&ProcessorLoad);
```

See Also

`SceCodecEnginePmonProcessorLoad`, `sceCodecEnginePmonStart()`, `sceCodecEnginePmonStop()`, `sceCodecEnginePmonReset()`

sceCodecEnginePmonReset

Reset Codec Engine performance measurement results

Definition

```
#include <codecengine.h>
SceInt32 sceCodecEnginePmonReset (
        void
)
```

Arguments

None

Return Values

Value	Description
0 (SCE_OK)	Success
<0	Error

Description

Resets Codec Engine performance measurement results.
Unless this function is executed, measurement results over multiple segments will be accumulated.

Notes

This function is not multi-thread safe.

Examples

```
SceCodecEnginePmonProcessorLoad ProcessorLoad;

// Reset measurement results
sceCodecEnginePmonReset();

// Start measuring performance
sceCodecEnginePmonStart();

// do something

// Stop measuring performance
sceCodecEnginePmonStop();

// Obtain Codec Engine processor load
ProcessorLoad.size = sizeof(ProcessorLoad);
sceCodecEnginePmonGetProcessorLoad(&ProcessorLoad);
```

See Also

SceCodecEnginePmonProcessorLoad, sceCodecEnginePmonGetProcessorLoad()

Constants

Error Codes

List of error codes returned by libcodecengine

Definition

Value	(Number)	Description
SCE_CODECENGINE_ERROR_INVALID_POINTER	0x80600000	Invalid pointer was specified to an argument
SCE_CODECENGINE_ERROR_INVALID_SIZE	0x80600001	Invalid size was specified to an argument
SCE_CODECENGINE_ERROR_INVALID_ALIGNMENT	0x80600002	Alignment size specified with argument is invalid
SCE_CODECENGINE_ERROR_NOT_PHYSICALLY_CONTIGUOUS	0x80600003	Memory specified with argument is not physically continuous
SCE_CODECENGINE_ERROR_INVALID_RANGE	0x80600004	Memory area specified with argument is invalid
SCE_CODECENGINE_ERROR_INVALID_HEAP	0x80600005	Invalid heap
SCE_CODECENGINE_ERROR_HEAP_BROKEN	0x80600006	Broken heap
SCE_CODECENGINE_ERROR_NO_MORE_ENTRY	0x80600007	Entry limit exceeded
SCE_CODECENGINE_ERROR_INVALID_MEMTYPE	0x80600008	Memory specified with argument is not mapped as cache-disabled
SCE_CODECENGINE_ERROR_INVALID_VALUE	0x80600009	Value specified with argument is invalid
SCE_CODECENGINE_ERROR_MEMORY_NOT_ALLOCATED	0x8060000A	Memory specified with argument is not allocated with `sceCodecEngineAllocMemoryFromUnmapMemBlock()`
SCE_CODECENGINE_ERROR_MEMORY_IN_USE	0x8060000B	Memory specified with argument is already in use
SCE_CODECENGINE_ERROR_MEMORY_NOT_IN_USE	0x8060000C	Memory specified with argument is not in use