

PlayStation®Edge Zlib Library Reference

Table of Contents

Preface.....	3
About This Document.....	4
Shared PPU/SPU Runtime Data Types.....	5
EdgeZlibDeflateQueueElement.....	6
EdgeZlibInflateQueueElement	7
EdgeZlibDeflateQHandle	9
EdgeZlibInflateQHandle	10
EdgeZlibDeflateTaskProcessing	11
EdgeZlibInflateTaskProcessing	12
EdgeZlibDeflateTaskProcessingStored	13
Shared PPU/SPU Functions.....	14
edgeZlibAddDeflateQueueElement	15
edgeZlibTryAddDeflateQueueElement	17
edgeZlibAddInflateQueueElement	19
edgeZlibTryAddInflateQueueElement	21
edgeZlibAddInflateQueueElementPartialCopyOut.....	23
edgeZlibTryAddInflateQueueElementPartialCopyOut	25
PPU Functions	27
edgeZlibGetDeflateQueueSize	28
edgeZlibGetInflateQueueSize	29
edgeZlibCreateDeflateQueue	30
edgeZlibCreateInflateQueue	31
edgeZlibShutdownDeflateQueue	32
edgeZlibShutdownInflateQueue	33
edgeZlibGetDeflateTaskContextSaveSize	34
edgeZlibGetInflateTaskContextSaveSize.....	35
edgeZlibCreateDeflateTask	36
edgeZlibCreateInflateTask	37
SPU Functions	38
edgeZlibDeflateRawData	39
edgeZlibInflateRawData.....	40
edgeZlibFetchAndDeflateRawData	41
edgeZlibFetchAndInflateRawData	43

Preface

About This Document

Purpose

This document provides an API reference for the Zlib component of the Edge library. Use this component to efficiently perform decompression or movement of data through the SPUs.

Audience and Prerequisites

This document was written for PlayStation®3 developers who want to write high-performance applications for the PlayStation®3. It is assumed that such developers have familiarity with the following:

- C and C++
- PlayStation®3 hardware
- SCE standard library functions

Related Documentation

In combination with this reference, the following documents provide complete usage and reference information about the Edge library:

- *PlayStation®Edge Library Overview*
- *PlayStation®Edge Geometry Library: Quick Start*
- *PlayStation®Edge Geometry Library Reference*
- *PlayStation®Edge Geometry Library for Offline Tool: Reference*
- *PlayStation®Edge Animation Library Reference*
- *PlayStation®Edge Animation Library for Offline Tool: Reference*
- *PlayStation®Edge LZMA Library Reference*
- *PlayStation®Edge LZO Library Reference*
- *PlayStation®Edge DXT Library Reference*
- *PlayStation®Edge Post Library Reference*

Typographic Conventions

This document uses the following typographic conventions:

Convention	Meaning
<code>fixed-width font</code>	Indicates programming code and literals, such as processing instructions, register names, data types, events, and file names. Also indicates function, structure and macro names.
<code>fixed-width font + bold</code>	In structure/function definitions only, indicates names of structures and functions.
<i>fixed-width font + italics</i>	Indicates arguments, parameters, and variables.
blue + underlined text	Indicates a hyperlink (blue displays in color printers or online only).

Shared PPU/SPU Runtime Data Types

EdgeZlibDeflateQueueElement

An entry in the queue that tells the Deflate Task to compress a single segment of data.

Definition

```
#include <edge/zlib/edgezlib_deflate_queue_element.h>
typedef struct EdgeZlibDeflateQueueElement
{
    uint32_t m_eaInputUncompressedData;
    uint32_t m_eaOutputCompressedData;
    uint32_t m_uncompressedSize;
    uint32_t m_maxCompressedOutputSize;
    uint32_t m_eaOutputCompressedSize;
    uint32_t m_eaWorkToDoCounter;
    uint32_t m_eaEventFlag;
    uint16_t m_eventFlagBits;
    uint16_t m_compressionLevel;
    uint16_t m_pad8;
} EdgeZlibDeflateQueueElement __attribute__((aligned(16)));
```

Members

<i>m_eaInputUncompressedData</i>	Effective address of input uncompressed data.
<i>m_eaOutputCompressedData</i>	Effective address where compressed data will be placed.
<i>m_uncompressedSize</i>	Size of uncompressed data.
<i>m_maxCompressedOutputSize</i>	Maximum size of output buffer for compressed data.
<i>m_eaOutputCompressedSize</i>	Effective address where SPU will return the size of the outputted data with the high bit set if the data was compressed.
<i>m_eaWorkToDoCounter</i>	Effective address of counter to atomically decrement when compression is done. May be NULL.
<i>m_eaEventFlag</i>	The 2 least significant bits give a flag of type EdgeZlibDeflateTaskProcessing . Effective address of the event flag. May be NULL.
<i>m_eventFlagBits</i>	The event flag will be set to this value after decompression is completed.
<i>m_compressionLevel</i>	Compression level (1 - fastest, 9 - smallest)
<i>m_pad8</i>	Ignored.

Description

This structure contains information about one segment of data. Each entry will cause a Deflate Task to compress this segment.

Notes

The address of the compressed and uncompressed data can have any alignment.

See Also

[edgeZlibAddDeflateQueueElement](#), [edgeZlibCreateDeflateQueue](#)

EdgeZlibInflateQueueElement

An entry in the queue that tells the Inflate Task to decompress or move a single segment of data.

Definition

```
#include <edge/zlib/edgezlib_inflate_queue_element.h>
typedef struct EdgeZlibInflateQueueElement
{
    uint32_t m_eaCompressed;
    uint32_t m_eaUncompressed;
    uint32_t m_compressedSize;
    uint32_t m_outputUncompPartialBuffSize;
    uint32_t m_eaWorkToDoCounter;
    uint32_t m_eaEventFlag;
    uint16_t m_eventFlagBits;
    uint16_t m_outputUncompSkipBeginSize;
    uint16_t m_outputUncompSkipEndSize;
    uint16_t m_pad16;
} EdgeZlibInflateQueueElement __attribute__((aligned(16)));
```

Members

<i>m_eaCompressed</i>	Effective address of compressed data.
<i>m_eaUncompressed</i>	Effective address where uncompressed data will be placed.
<i>m_compressedSize</i>	Size of compressed data.
<i>m_outputUncompPartialBuffSize</i>	Size of the uncompressed data to be DMAed out.
<i>m_eaWorkToDoCounter</i>	Effective address of counter to atomically decrement when decompression is done. May be NULL.
<i>m_eaEventFlag</i>	Least significant bit is a flag that indicates if the segment was stored compressed (1) or uncompressed (0). Effective address of the event flag. May be NULL.
<i>m_eventFlagBits</i>	The event flag will be set to this value after decompression is completed.
<i>m_outputUncompSkipBeginSize</i>	Do not output the first <i>N</i> bytes of the uncompressed output.
<i>m_outputUncompSkipEndSize</i>	Do not output the last <i>N</i> bytes of the uncompressed output.
<i>m_pad16</i>	Ignored.

Description

This structure contains information about one segment of data. Each entry will cause an Inflate Task to decompress or move this memory.

Notes

The address of the compressed and uncompressed data can have any alignment.

The sum of *m_outputUncompSkipBeginSize*, *m_outputUncompPartialBuffSize*, and *m_outputUncompSkipEndSize* gives the expected size that the compressed data will decompress to. If this value is incorrect, the SPU code will assert.

See Also

[edgeZlibAddInflateQueueElement](#), [edgeZlibAddInflateQueueElementPartialCopyOut](#),
[edgeZlibCreateInflateQueue](#)

EdgeZlibDeflateQHandle

A handle for the Deflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
typedef void* EdgeZlibDeflateQHandle;
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
typedef uint32_t EdgeZlibDeflateQHandle;
```

Description

This is a handle for the Deflate Queue in main memory.

See Also

[EdgeZlibDeflateQueueElement](#), [edgeZlibCreateDeflateQueue](#)

EdgeZlibInflateQHandle

A handle for the Inflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
typedef void* EdgeZlibInflateQHandle;
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
typedef uint32_t EdgeZlibInflateQHandle;
```

Description

This is a handle for the Inflate Queue in main memory.

See Also

[EdgeZlibInflateQueueElement](#), [edgeZlibCreateInflateQueue](#)

EdgeZlibDeflateTaskProcessing

Specifies whether a segment is to be stored compressed, or if the smaller of the compressed and uncompressed data is to be stored.

Definition

```
#include <edge/zlib/edgezlib_deflate_queue_element.h>

typedef enum EdgeZlibDeflateTaskProcessing
{
    kEdgeZlibDeflateTask_DeflateStoreCompressed = 0,
    kEdgeZlibDeflateTask_DeflateStoreSmallest = 1,
    kEdgeZlibDeflateTask_DeflateStoreCompressedWithHeader = 2,
} EdgeZlibDeflateTaskProcessing;
```

Members

<i>kEdgeZlibDeflateTask_DeflateStoreCompressed</i>	Always store compressed data.
<i>kEdgeZlibDeflateTask_DeflateStoreSmallest</i>	Store either compressed or uncompressed data, whichever is smaller.
<i>kEdgeZlibDeflateTask_DeflateStoreCompressedWithHeader</i>	Always store compressed data, with the zlib 2-byte header and 4-byte footer added.

Description

This is used to declare whether to add a zlib 2-byte header and 4-byte footer to the compressed data, and, in the case where compression does not reduce the size, whether to store the smaller of the compressed data or the uncompressed data (without the header and footer).

See Also

[edgeZlibAddDeflateQueueElement](#)

EdgeZlibInflateTaskProcessing

Specifies whether a segment is copied or decompressed.

Definition

```
#include <edge/zlib/edgezlib_inflate_queue_element.h>
typedef enum EdgeZlibInflateTaskProcessing
{
    kEdgeZlibInflateTask_Memcpy = 0,
    kEdgeZlibInflateTask_Inflate = 1,
} EdgeZlibInflateTaskProcessing;
```

Members

<i>kEdgeZlibInflateTask_Memcpy</i>	Copy memory from one location to another.
<i>kEdgeZlibInflateTask_Inflate</i>	Decompress memory from one location to another.

Description

This is used to declare whether the segment is to be copied or decompressed by the Inflate Task.

See Also

[edgeZlibAddInflateQueueElement](#), [edgeZlibAddInflateQueueElementPartialCopyOut](#)

EdgeZlibDeflateTaskProcessingStored

The Deflate Task specifies whether the output data was stored compressed or uncompressed.

Definition

```
#include <edge/zlib/edgezlib_deflate_queue_element.h>

typedef enum EdgeZlibDeflateTaskProcessingStored
{
    kEdgeZlibDeflateTask_UncompressedWasStored = 0x00000000,
    kEdgeZlibDeflateTask_CompressedWasStored = 0x80000000,
} EdgeZlibDeflateTaskProcessingStored;
```

Members

<i>kEdgeZlibDeflateTask_UncompressedWasStored</i>	Uncompressed data was stored.
<i>kEdgeZlibDeflateTask_CompressedWasStored</i>	Compressed data was stored (with or without optional zlib header and footer).

Description

When the SPU writes out the output size, the high bit is used to declare whether the data was stored compressed or uncompressed. The compressed data may have an optional zlib 2-byte header and 4-byte footer depending on whether the user requested it.

See Also

[edgeZlibAddDeflateQueueElement](#)

Shared PPU/SPU Functions

edgeZlibAddDeflateQueueElement

Adds a new piece of work to the Deflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
void edgeZlibAddDeflateQueueElement
(
    EdgeZlibDeflateQHandle handle,
    const void* eaInputUncompressedData,
    uint32_t uncompressedSize,
    void* eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t* eaOutputCompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    uint32_t level,
    EdgeZlibDeflateTaskProcessing processing,
)
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
void edgeZlibAddDeflateQueueElement
(
    EdgeZlibDeflateQHandle handle,
    uint32_t eaInputUncompressedData,
    uint32_t uncompressedSize,
    uint32_t eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t eaOutputCompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    uint32_t level,
    EdgeZlibDeflateTaskProcessing processing,
    uint32_t dmaTag
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Deflate Queue that the entry will be pushed onto.
<i>eaInputUncompressedData</i>	The Effective Address of the input data that is to be compressed. This should be a pointer to the raw data without any header.
<i>uncompressedSize</i>	The size of the uncompressed input data.
<i>eaOutputCompressedData</i>	The Effective Address of the output buffer for the compressed data.
<i>maxCompressedOutputSize</i>	The maximum space available for the compressed data buffer.

<i>eaOutputCompressedSize</i>	The Effective Address of the <code>uint32_t</code> into which the output compressed size will be written. The top bit of the written counter indicates if the data was stored compressed or if the uncompressed original data was chosen for storing.
<i>eaWorkToDoCounter</i>	A counter in main memory that will be atomically decremented after this item has been decompressed. If the SPU has an error with the compressed data, the high bit will be set as an error flag. Can be NULL.
<i>eaEventFlag</i>	The event flag to set. Can be NULL.
<i>eventFlagBits</i>	The event flag will be set to this value after compression is completed.
<i>level</i>	Choose the compression level between 0 and 9: 0 - no compression 1 - fastest 9 - smallest
<i>processing</i>	Choose whether compressed data will have optional zlib 2-byte header and 4-byte footer added, and whether the smaller of the compressed and uncompressed data is to be stored.
<i>dmaTag</i>	(SPU only) The DMA tag to be used for DMAs performed inside this function.

Return Values

None.

Description

Adds a new piece of work to the Deflate Queue. The work added to this queue will be taken by one of the Deflate Tasks at the next opportunity.

Notes

If the queue is full (that is, if it has reached `maxNumQueueEntries`), and another item is pushed onto the queue, then this function will block until there is space.

The work-to-do-counter can be used to track completion of the processing on a collection of segments by initializing it to the number of segments being waited on. Then when each item completes it will decrement by one, so when the value reaches zero all segments will have been done. At this point the specified event flag will be set. Note that if the SPU has an error with compressed data it will set the high bit of the counter.

If *eaWorkToDoCounter* is NULL, but *eaEventFlag* is valid, the event flag will be set after this segment is done.

See Also

[EdgeZlibDeflateQueueElement](#), [edgeZlibCreateDeflateTask](#), [edgeZlibTryAddDeflateQueueElement](#)

edgeZlibTryAddDeflateQueueElement

Tries to add a new piece of work to the Deflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
bool edgeZlibTryAddDeflateQueueElement
(
    EdgeZlibDeflateQHandle handle,
    const void* eaInputUncompressedData,
    uint32_t uncompressedSize,
    void* eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t* eaOutputCompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    uint32_t level,
    EdgeZlibDeflateTaskProcessing processing,
)
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
bool edgeZlibTryAddDeflateQueueElement
(
    EdgeZlibDeflateQHandle handle,
    uint32_t eaInputUncompressedData,
    uint32_t uncompressedSize,
    uint32_t eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t eaOutputCompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    uint32_t level,
    EdgeZlibDeflateTaskProcessing processing,
    uint32_t dmaTag
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Deflate Queue that the entry will be pushed onto.
<i>eaInputUncompressedData</i>	The Effective Address of the input data that is to be compressed. This should be a pointer to the raw data without any header.
<i>uncompressedSize</i>	The size of the uncompressed input data.
<i>eaOutputCompressedData</i>	The Effective Address of the output buffer for the compressed data.
<i>maxCompressedOutputSize</i>	The maximum space available for the compressed data.

<i>eaOutputCompressedSize</i>	The Effective Address of the <code>uint32_t</code> into which the output compressed size will be written. The top bit of the written counter indicates if the data was stored compressed or if the uncompressed original data was chosen for storing.
<i>eaWorkToDoCounter</i>	A counter in main memory that will be atomically decremented after this item has been decompressed. If the SPU has an error with the compressed data, the high bit will be set as an error flag. Can be NULL.
<i>eaEventFlag</i>	The event flag to set. Can be NULL.
<i>eventFlagBits</i>	The event flag will be set to this value after compression is completed.
<i>level</i>	Choose the compression level between 0 and 9: 0 - no compression 1 - fastest 9 - smallest
<i>processing</i>	Choose whether compressed data will have optional zlib 2-byte header and 4-byte footer added, and whether the smaller of the compressed and uncompressed data is to be stored.
<i>dmaTag</i>	(SPU only.) The DMA tag to be used for DMAs performed inside this function.

Return Values

`true` if the item is added to the deflate queue; `false` if it could not be added at this time.

Description

Tries to add a new piece of work to the Deflate Queue. If added, the work added to this queue will be taken by one of the Deflate Tasks at the next opportunity.

Notes

If the queue is full (that is, if it has reached `maxNumQueueEntries`), and another item is pushed onto the queue, then this function will return `false` immediately.

The work-to-do-counter can be used to track completion of the processing on a collection of segments by initializing it to the number of segments being waited on. Then when each item completes it will decrement by one, so when the value reaches zero all segments will have been done. At this point the specified event flag will be set. Note that if the SPU has an error with compressed data it will set the high bit of the counter.

If *eaWorkToDoCounter* is NULL, but *eaEventFlag* is valid, the event flag will be set after this segment is done.

See Also

[EdgeZlibDeflateQueueElement](#), [edgeZlibCreateDeflateTask](#), [edgeZlibAddDeflateQueueElement](#)

edgeZlibAddInflateQueueElement

Adds a new piece of work to the Inflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
void edgeZlibAddInflateQueueElement
(
    EdgeZlibInflateQHandle handle,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing
)
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
void edgeZlibAddInflateQueueElement
(
    EdgeZlibInflateQHandle handle,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing,
    uint32_t dmaTag
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Inflate Queue that the entry will be pushed onto.
<i>eaInputCompressedData</i>	The Effective Address of the input data that is to be decompressed. This should be a pointer to the raw data without any header.
<i>compressedSize</i>	The size of the compressed input data.
<i>eaOutputUncompressed</i>	The Effective Address of the output buffer for the uncompressed data.
<i>expectedUncompressedSize</i>	The expected size of the uncompressed data. The SPU will assert if this value is incorrect.
<i>eaWorkToDoCounter</i>	A counter in main memory that will be atomically decremented after this item has been decompressed. Can be NULL.

<i>eaEventFlag</i>	The event flag to set. Can be NULL.
<i>eventFlagBits</i>	The event flag will be set to this value after decompression is completed.
<i>processing</i>	Choose what kind of processing the task has to do on the data. Can perform decompression, or merely move the raw data from <i>eaInputCompressedData</i> to <i>eaOutputUncompressedData</i> .
<i>dmaTag</i>	(SPU only.) The DMA tag to be used for DMAs performed inside this function.

Return Values

None.

Description

Adds a new piece of work to the Inflate Queue. The work added to this queue will be taken by one of the Inflate Tasks at the next opportunity.

Notes

If the queue is full (that is, if it has reached `maxNumQueueEntries`), and another item is pushed onto the queue, then this function will block until there is space.

The work-to-do-counter can be used to track completion of the processing on a collection of segments by initializing it to the number of segments being waited on. Then when each item completes it will decrement by one, and so when the value reaches zero all segments will have been done. At this point the specified event flag will be set. Note that if the SPU has an error with compressed data it will set the high bit of the counter.

If *eaWorkToDoCounter* is NULL, but *eaEventFlag* is valid, the event flag will be set after this segment is done.

If the data was stored uncompressed and merely needs to be moved to its destination, the Inflate Task can be used for this by passing the appropriate value for performing only a memory copy.

This function expects a pointer to the raw compressed data without any header.

See Also

[EdgeZlibInflateQueueElement](#), [edgeZlibAddInflateQueueElementPartialCopyOut](#), [edgeZlibCreateInflateTask](#)

edgeZlibTryAddInflateQueueElement

Tries to add a new piece of work to the Inflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
bool edgeZlibTryAddInflateQueueElement
(
    EdgeZlibInflateQHandle handle,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing,
)
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
bool edgeZlibTryAddInflateQueueElement
(
    EdgeZlibInflateQHandle handle,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing,
    uint32_t dmaTag
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Inflate Queue that the entry will be pushed onto.
<i>eaInputCompressedData</i>	The Effective Address of the input data which is to be decompressed.
<i>compressedSize</i>	This should be a pointer to the raw data without any header.
<i>eaOutputUncompressed</i>	The size of the compressed input data.
<i>expectedUncompressedSize</i>	The Effective Address of the output buffer for the uncompressed data.
	The expected size of the uncompressed data.
	The SPU will assert if this value is incorrect.

<i>eaWorkToDoCounter</i>	A counter in main memory that will be atomically decremented after this item has been decompressed. If the SPU has an error with the compressed data, the high bit of the counter will be set as an error flag. Can be NULL.
<i>eaEventFlag</i>	The event flag to set. Can be NULL.
<i>eventFlagBits</i>	The value to set to the event flag.
<i>processing</i>	Choose what kind of processing the task has to do on the data. Can perform decompression, or merely move the raw data from <i>eaInputCompressedData</i> to <i>eaOutputUncompressedData</i> .
<i>dmaTag</i>	The DMA tag to be used for DMAs performed inside this function (only on SPU).

Return Values

`true` if the item was successfully added to the queue; `false` if it failed to be added.

Description

Tries to add a new piece of work to the Inflate Queue. The work added to this queue will be taken by one of the Inflate Tasks at the next opportunity .

Notes

If the item is added, this function will return `true`. If the queue is full (that is, if *maxNumQueueEntries* has been reached), and another item is pushed onto the queue, this function will return `false`.

The work-to-do-counter can be used to track completion of the processing on a collection of segments by initializing it to the number of segments being waited on. Then when each item completes it will decrement by one, and so when the value reaches zero all segments will have been done. At this point the specified event flag will be set. Note that if the SPU has an error with the compressed data, it will set the high bit of the counter.

If *eaWorkToDoCounter* is NULL, but *eaEventFlag* is valid, the event flag will be set after this segment is done.

If the data was stored uncompressed and merely needs moving to its destination, the Inflate Task can be used for this by passing the appropriate value for *processing*.

This function expects a pointer to the raw compressed data without any header.

See Also

[EdgeZlibInflateQueueElement](#), [edgeZlibAddInflateQueueElementPartialCopyOut](#), [edgeZlibCreateInflateTask](#)

edgeZlibAddInflateQueueElementPartialCopyOut

Adds a new piece of work to the Inflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
void edgeZlibAddInflateQueueElementPartialCopyOut
(
    EdgeZlibInflateQHandle handle,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompPartialBuff,
    uint16_t outputUncompSkipBeginSize,
    uint32_t outputUncompPartialBuffSize,
    uint16_t outputUncompSkipEndSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing,
)
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
void edgeZlibAddInflateQueueElementPartialCopyOut
(
    EdgeZlibInflateQHandle handle,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompPartialBuff,
    uint16_t outputUncompSkipBeginSize,
    uint32_t outputUncompPartialBuffSize,
    uint16_t outputUncompSkipEndSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing,
    uint32_t dmaTag
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Inflate Queue that the entry will be pushed onto.
<i>eaInputCompressedData</i>	The Effective Address of the input data which is to be decompressed.
<i>compressedSize</i>	This should be a pointer to the raw data without any header.
<i>eaOutputUncompPartialBuff</i>	The size of the compressed input data.
	The Effective Address of the output buffer for the uncompressed data.

<i>outputUncompSkipBeginSize</i>	Do not output the first N bytes of the uncompressed output.
<i>outputUncompPartialBuffSize</i>	The size of the uncompressed data that will be DMAed out.
<i>outputUncompSkipEndSize</i>	Do not output the last N bytes of the uncompressed output.
<i>eaWorkToDoCounter</i>	A counter in main memory that will be atomically decremented after this item has been decompressed. Can be NULL.
<i>eaEventFlag</i>	The event flag to set. Can be NULL.
<i>eventFlagBits</i>	The value to set to the event flag
<i>processing</i>	Choose what kind of processing the task has to do on the data. Can perform decompression, or merely move the raw data from <i>eaInputCompressedData</i> to <i>eaOutputUncompressedData</i> .
<i>dmaTag</i>	The DMA tag to be used for DMAs performed inside this function (only on SPU).

Return Values

None.

Description

Adds a new piece of work to the Inflate Queue. The work added to this queue will be taken by one of the Inflate Tasks at the next opportunity.

Notes

If the queue is full (that is, if it has reached `maxNumQueueEntries`), and another item is pushed onto the queue, this function will block until there is space.

The work-to-do-counter can be used to track completion of the processing on a collection of segments by initializing it to the number of segments being waited on. Then when each item completes it will decrement by one, and so when the value reaches zero all segments will have been done. At this point the specified event flag will be set. Note that if the SPU has an error with the compressed data, it will set the high bit of the counter.

If *eaWorkToDoCounter* is NULL, but *eaEventFlag* is valid, the event flag will be set after this segment is done.

If the data was stored uncompressed and merely needs to be moved to its destination, the Inflate Task can be used for this by passing the appropriate value for 'processing'.

This function expects a pointer to the raw compressed data without any header.

This function is very similar to [edgeZlibAddInflateQueueElement](#). The additional parameters *skipOutputBeginSize* and *skipOutputEndSize* exist so that decompression of a segment can be done, but only a portion of the output may need to be written.

The sum of the *outputUncompSkipBeginSize*, *outputUncompPartialBuffSize*, and *outputUncompSkipEndSize* gives the expected uncompressed size of the compressed data. The SPU will assert if this value is incorrect.

See Also

[EdgeZlibInflateQueueElement](#), [edgeZlibAddInflateQueueElement](#), [edgeZlibCreateInflateTask](#)

edgeZlibTryAddInflateQueueElementPartialCopyOut

Tries to add a new piece of work to the Inflate Queue.

Definition (on PPU)

```
#include <edge/zlib/edgezlib_ppu.h>
bool edgeZlibTryAddInflateQueueElementPartialCopyOut
(
    EdgeZlibInflateQHandle handle,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompPartialBuff,
    uint16_t outputUncompSkipBeginSize,
    uint32_t outputUncompPartialBuffSize,
    uint16_t outputUncompSkipEndSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing,
)
```

Definition (on SPU)

```
#include <edge/zlib/edgezlib_spu.h>
bool edgeZlibTryAddInflateQueueElementPartialCopyOut
(
    EdgeZlibInflateQHandle handle,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompPartialBuff,
    uint16_t outputUncompSkipBeginSize,
    uint32_t outputUncompPartialBuffSize,
    uint16_t outputUncompSkipEndSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeZlibInflateTaskProcessing processing,
    uint32_t dmaTag
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Inflate Queue that the entry will be pushed onto.
<i>eaInputCompressedData</i>	The Effective Address of the input data which is to be decompressed.
<i>compressedSize</i>	This should be a pointer to the raw data without any header.
<i>eaOutputUncompPartialBuff</i>	The size of the compressed input data.
	The Effective Address of the output buffer for the uncompressed data.

<i>outputUncompSkipBeginSize</i>	Do not output the first N bytes of the uncompressed output.
<i>outputUncompPartialBuffSize</i>	The size of the uncompressed data which will be DMAed out.
<i>outputUncompSkipEndSize</i>	Do not output the last N bytes of the uncompressed output.
<i>eaWorkToDoCounter</i>	A counter in main memory that will be atomically decremented after this item has been decompressed. If the SPU has an error with the compressed data, it will set the high bit of the counter as an error flag. Can be NULL.
<i>eaEventFlag</i>	The event flag to set. Can be NULL.
<i>eventFlagBits</i>	The value to set to the event flag.
<i>processing</i>	Choose what kind of processing the task has to do on the data. Can perform decompression, or merely move the raw data from <i>eaInputCompressedData</i> to <i>eaOutputUncompressedData</i> .
<i>dmaTag</i>	The DMA tag to be used for DMAs performed inside this function (only on SPU).

Return Values

`true` if the item was successfully added to the queue; `false` if it failed to be added.

Description

Tries to add a new piece of work to the Inflate Queue. The work added to this queue will be taken by one of the Inflate Tasks at the next opportunity.

Notes

If the item is added, then this function will return `true`. If the queue is full (that is, if *maxNumQueueEntries* has been reached), and another item is pushed onto the queue, then this function will return `false`.

The work-to-do-counter can be used to track completion of the processing on a collection of segments by initializing it to the number of segments being waited on. Then when each item completes it will decrement by one, and so when the value reaches zero all segments will have been done. At this point the specified event flag will be set. Note that if the SPU has an error with the compressed data, it will set the high bit of the counter.

If *eaWorkToDoCounter* is NULL, but *eaEventFlag* is valid, the event flag will be set after this segment is done.

If the data was stored uncompressed and merely needs moving to its destination, the Inflate Task can be used for this by passing the appropriate value for *processing*.

This function expects a pointer to the raw compressed data without any header.

This function is very similar to [edgeZlibAddInflateQueueElement](#). The additional parameters *skipOutputBeginSize* and *skipOutputEndSize* exist so that decompression of a segment can be done, but only a portion of the output may need to be written.

The sum of the *outputUncompSkipBeginSize*, *outputUncompPartialBuffSize* and *outputUncompSkipEndSize* gives the expected uncompressed size of the compressed data. The SPU will assert if this value is incorrect.

See Also

[EdgeZlibInflateQueueElement](#), [edgeZlibAddInflateQueueElement](#), [edgeZlibCreateInflateTask](#)

PPU Functions

edgeZlibGetDeflateQueueSize

Returns the required buffer size needed for a Deflate Queue.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
uint32_t edgeZlibGetDeflateQueueSize
(
    uint32_t maxNumQueueEntries
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

maxNumQueueEntries The maximum number of entries the queue will hold at one time (maximum: 32767)

Return Values

Returns the required buffer size needed.

Description

Returns the required buffer size needed by a Deflate Queue to be able to hold the specified number of elements.

Notes

The allocated buffer must be 128-byte aligned.

See Also

[EdgeZlibDeflateQueueElement](#), [edgeZlibCreateDeflateQueue](#), [edgeZlibCreateDeflateTask](#)

edgeZlibGetInflateQueueSize

Returns the required buffer size needed for an Inflate Queue.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
uint32_t edgeZlibGetInflateQueueSize
(
    uint32_t maxNumQueueEntries
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

maxNumQueueEntries The maximum number of entries the queue will hold at one time (maximum: 32767)

Return Values

Returns the required buffer size needed.

Description

Returns the required buffer size needed by an Inflate Queue to be able to hold the specified number of elements.

Notes

The allocated buffer must be 128-byte aligned.

See Also

[EdgeZlibInflateQueueElement](#), [edgeZlibCreateInflateQueue](#), [edgeZlibCreateInflateTask](#)

edgeZlibCreateDeflateQueue

Creates a Deflate Queue.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
EdgeZlibDeflateQHandle edgeZlibCreateDeflateQueue
(
    CellSpurs *pSpurs,
    uint32_t maxNumQueueEntries,
    void *pBuffer,
    uint32_t bufferSize
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>pSpurs</i>	Pointer to the SPURS instance that this Deflate Queue will be associated with.
<i>maxNumQueueEntries</i>	The maximum number of entries this queue will hold at one time (maximum: 32767).
<i>pBuffer</i>	Pointer to the buffer in main memory to be used for this Deflate Queue. Must be aligned to 128 bytes.
<i>bufferSize</i>	The size of the provided buffer in main memory. The required size of this buffer for a given number of queue elements can be queried by calling edgeZlibGetDeflateQueueSize() .

Return Values

The [EdgeZlibDeflateQHandle](#) of the created Deflate Queue.

Description

Creates a Deflate Queue. This will hold the list of work (segments to compress) that is queued up for the Deflate Tasks to work on.

Notes

The Deflate Queue is a FIFO and will stall as necessary if work is pushed onto a full queue, so the queue size can safely be lower than the actual maximum number of elements needed.

See Also

[EdgeZlibDeflateQueueElement](#), [edgeZlibAddDeflateQueueElement](#), [edgeZlibGetDeflateQueueSize](#), [edgeZlibShutdownDeflateQueue](#)

edgeZlibCreateInflateQueue

Creates an Inflate Queue.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
EdgeZlibInflateQHandle edgeZlibCreateInflateQueue
(
    CellSpurs *pSpurs,
    uint32_t maxNumQueueEntries,
    void* pBuffer,
    uint32_t bufferSize
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>pSpurs</i>	Pointer to the SPURS instance that this Inflate Queue will be associated with.
<i>maxNumQueueEntries</i>	The maximum number of entries this queue will hold at one time (maximum: 32767).
<i>pBuffer</i>	Pointer to the buffer in main memory to be used for this Inflate Queue. Must be aligned to 128 bytes.
<i>bufferSize</i>	The size of the provided buffer in main memory. The required size of this buffer for a given number of queue elements can be queried by calling edgeZlibGetInflateQueueSize() .

Return Values

The [EdgeZlibInflateQHandle](#) of the created Inflate Queue.

Description

Creates an Inflate Queue. This will hold the list of work (segments to decompress or move) that is queued up for the Inflate Tasks to work on.

Notes

The Inflate Queue is a FIFO and will stall as necessary if work is pushed onto a full queue, so the queue size can safely be lower than the actual maximum number of elements needed.

See Also

[EdgeZlibInflateQueueElement](#), [edgeZlibAddInflateQueueElement](#), [edgeZlibTryAddInflateQueueElementPartialCopyOut](#), [edgeZlibGetInflateQueueSize](#), [edgeZlibShutdownInflateQueue](#), [edgeZlibCreateInflateTask](#)

edgeZlibShutdownDeflateQueue

Shuts down the Deflate Queue.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
void edgeZlibShutdownDeflateQueue
(
    EdgeZlibDeflateQHandle handle
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Deflate Queue to shutdown
---------------	---

Return Values

None.

Description

Shuts down the Deflate Queue.

See Also

[edgeZlibCreateDeflateQueue](#)

edgeZlibShutdownInflateQueue

Shuts down the Inflate Queue.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
void edgeZlibShutdownInflateQueue
(
    EdgeZlibInflateQHandle handle
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>handle</i>	The handle of the Inflate Queue to shutdown
---------------	---

Return Values

None.

Description

Shuts down the Inflate Queue.

See Also

[edgeZlibCreateInflateQueue](#)

edgeZlibGetDeflateTaskContextSaveSize

Returns the required buffer size needed by one Deflate Task for storing its context.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
uint32_t edgeZlibGetDeflateTaskContextSaveSize( void )
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

(none)

Return Values

The size of the buffer needed for storing the context data of one Deflate Task.

Description

This function returns the required buffer size that the library needs for storing the context of a single Deflate Task.

Notes

The allocated buffer must be 16-byte aligned.

See Also

[edgeZlibCreateDeflateTask](#)

edgeZlibGetInflateTaskContextSaveSize

Returns the required buffer size needed by one Inflate Task for storing its context.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
uint32_t edgeZlibGetInflateTaskContextSaveSize( void )
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

(none)

Return Values

The size of the buffer needed for storing the context data of one Inflate Task.

Description

This function returns the required buffer size that the library needs for storing the context of a single Inflate Task.

Notes

The allocated buffer must be 16-byte aligned.

See Also

[edgeZlibCreateInflateTask](#)

edgeZlibCreateDeflateTask

Creates a SPURS Task for performing compression on one SPU.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
CellSpursTaskId edgeZlibCreateDeflateTask
(
    CellSpursTaskset* pTaskSet,
    void* pTaskContext,
    EdgeZlibDeflateQHandle handle
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>pTaskSet</i>	Pointer to the SPURS Taskset that this Deflate Task should be attached to.
<i>pTaskContext</i>	Pointer to the main memory buffer that the task uses for storing its context. The required size of this buffer can be queried by calling edgeZlibGetDeflateTaskContextSaveSize() .
<i>handle</i>	The handle for the Deflate Queue that this task will be pulling from.

Return Values

The `CellSpursTaskId` of the created task.

Description

When work exists, this task will pull work off the Deflate Queue.

If there is no work to do, the task will sleep, in which case it will store its context in the location specified by *pTaskContext*.

Notes

Create one Deflate Task for each SPU you want to run on. So, if you want compression to be able to run in parallel on six SPUs (and your SPURS Instance has six SPUs in it), you should create six Deflate Tasks, all in the same taskset and all working on the same Deflate Queue.

The necessary size for the buffer pointed to by *pTaskContext* may be queried by calling [edgeZlibGetDeflateTaskContextSaveSize\(\)](#).

See Also

[edgeZlibAddDeflateQueueElement](#), [edgeZlibGetDeflateQueueSize](#), [edgeZlibCreateDeflateQueue](#), [edgeZlibShutdownDeflateQueue](#), [edgeZlibGetDeflateTaskContextSaveSize](#)

edgeZlibCreateInflateTask

Creates a SPURS Task for performing decompression on one SPU.

Definition

```
#include <edge/zlib/edgezlib_ppu.h>
CellSpursTaskId edgeZlibCreateInflateTask
(
    CellSpursTaskset* pTaskSet,
    void* pTaskContext,
    EdgeZlibInflateQHandle handle
)
```

Calling Conditions

Can be called from an interrupt handler.

Can be called from a thread (does not depend on interrupt-disabled or -enabled state).

Multithread safe.

Arguments

<i>pTaskSet</i>	Pointer to the SPURS Taskset that this Inflate Task should be attached to.
<i>pTaskContext</i>	Pointer to the main memory buffer that the task uses for storing its context. The required size of this buffer can be queried by calling edgeZlibGetInflateTaskContextSaveSize .
<i>handle</i>	The handle for the Inflate Queue that this task will be pulling from.

Return Values

The `CellSpursTaskId` of the created task.

Description

When work exists, this task will pull work off the Inflate Queue.

If there is no work to do, the task will sleep, in which case it will store its context in the location specified by *pTaskContext*.

Notes

Create one Inflate Task for each SPU you want to run on. So, if you want decompression to be able to run in parallel on six SPUs (and your SPURS Instance has six SPUs in it) then you should create six Inflate Tasks, all in the same task set, and all working on the same Inflate Queue.

The necessary size for the buffer pointed to by *pTaskContext* may be queried by calling [edgeZlibGetInflateTaskContextSaveSize](#).

See Also

[edgeZlibAddInflateQueueElement](#), [edgeZlibTryAddInflateQueueElementPartialCopyOut](#), [edgeZlibGetInflateQueueSize](#), [edgeZlibCreateInflateQueue](#), [edgeZlibShutdownInflateQueue](#), [edgeZlibGetInflateTaskContextSaveSize](#)

SPU Functions

edgeZlibDeflateRawData

Compresses a buffer of uncompressed data.

Definition

```
#include <edge/zlib/edgezlib_spu.h>
extern int edgeZlibDeflateRawData
(
    const unsigned char* pUncompr,
    uint32_t uncompSize,
    unsigned char* pComprData,
    uint32_t maxCompressedDataSize,
    uint32_t* pOutputCompressedSize,
    uint32_t level,
);
```

Arguments

<i>pUncompr</i>	Pointer to where the uncompressed data will be read from in Local Store.
<i>uncompSize</i>	The size of the input compressed data that will be compressed.
<i>pComprData</i>	Pointer to where the compressed data will be written to in Local Store. This area will be filled with the raw compressed data without any header.
<i>maxCompressedDataSize</i>	Maximum size of the compressed data buffer.
<i>pOutputCompressedSize</i>	Returns the output size of the compressed data.
<i>level</i>	Choose the compression level between 0 and 9: 0 - no compression 1 - fastest 9 - smallest

Return Values

Zero on success; nonzero on failure.

Description

This function compresses a buffer of uncompressed data within Local Store (LS).

Notes

The input and output buffers are both in LS. Functions that call [edgeZlibDeflateRawData\(\)](#) are expected to provide the input data and deal with the output data. This function does not do any DMAs internally.

edgeZlibInflateRawData

Decompresses a buffer of compressed data.

Definition

```
#include <edge/zlib/edgezlib_spu.h>
extern int edgeZlibInflateRawData
(
    unsigned char* pUncompr,
    uint32_t expectedUncompSize,
    const unsigned char* pComprData,
    uint32_t comprDataSize
);
```

Arguments

<i>pUncompr</i>	Pointer to where the uncompressed data will be written to in Local Store.
<i>expectedUncompSize</i>	The expected output size of the uncompressed data. The output buffer pointed to by <i>pUncompr</i> must be at least this large.
<i>pComprData</i>	Pointer to where the compressed data will be read from in Local Store. This should be a pointer to the raw data without any header.
<i>comprDataSize</i>	Size of the compressed data buffer to be read.

Return Values

Zero on success; nonzero on failure.

Description

This function decompresses a buffer of compressed data within Local Store (LS).

Notes

The input and output buffers are both in LS. Functions that call [edgeZlibInflateRawData\(\)](#) are expected to provide the input data and deal with the output data. This function does not do any DMAs internally.

This function will assert if the *expectedUncompSize* disagrees with the actual uncompressed size.

This function expects a pointer to the raw compressed data without any header.

If any data error is encountered, this function will not assert (unless a conditional define is changed) and instead returns an error value to its caller.

edgeZlibFetchAndDeflateRawData

Compresses a buffer of uncompressed data in main memory to another main memory buffer.

Definition

```
#include <edge/zlib/edgezlib_spu.h>
extern int edgeZlibFetchAndDeflateRawData
(
    uint32_t eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t eaOutputSize,
    uint32_t eaInputUncompressedData,
    uint32_t uncompressedSize,
    uint32_t dmaTag,
    unsigned char* pLsInputTempBuffer,
    uint32_t inputTempBuffSize,
    unsigned char* pLsOutputTempBuffer,
    uint32_t outputTempBuffSize,
    uint32_t level,
    EdgeZlibDeflateTaskProcessing taskProcessing
);
```

Arguments

<i>eaOutputCompressedData</i>	The Effective Address of the output buffer for the compressed data.
<i>maxCompressedOutputSize</i>	The maximum space available for the compressed data.
<i>eaOutputSize</i>	The Effective Address of the <code>uint32_t</code> into which the output compressed size will be written. The top bit of the outputted size indicates whether the data was stored compressed or if the uncompressed original data was chosen for storing.
<i>eaInputUncompressedData</i>	The Effective Address of the input data that is to be compressed. This should be a pointer to the raw data without any header.
<i>uncompressedSize</i>	The size of the uncompressed input data.
<i>dmaTag</i>	The DMA tag to be used for DMAs performed inside this function.
<i>pLsInputTempBuffer</i>	Pointer to the temporary buffer in Local Store to be used for reading input data into. The size of this buffer is specified by <i>inputTempBuffSize</i> .
<i>inputTempBuffSize</i>	Size of the temporary buffer pointed to by <i>pLsInputTempBuffer</i> .
<i>pLsOutputTempBuffer</i>	Pointer to the temporary buffer in Local Store to be used for computing the output data into and sending out from. The size of this buffer is specified by <i>outputTempBuffSize</i> .
<i>outputTempBuffSize</i>	Size of the temporary buffer pointed to by <i>pLsOutputTempBuffer</i> .
<i>level</i>	Choose the compression level between 0 and 9: 0 - no compression 1 - fastest 9 - smallest
<i>taskProcessing</i>	Choose whether compressed data will have an optional zlib 2-byte header and 4-byte footer added, and whether the smaller of the compressed and uncompressed data is to be stored.

Return Values

Zero on success; nonzero on failure.

Description

This function compresses a buffer of uncompressed data from main memory. This function fetches data into LS and sends data out of LS as necessary.

Notes

This function requires two temporary buffers in LS. These can be specified as parameters from the calling function (*pLsInputTempBuffer* and *pLsOutputTempBuffer*).

edgeZlibFetchAndInflateRawData

Decompresses a buffer of compressed data in main memory to another main memory buffer.

Definition

```
#include <edge/zlib/edgezlib_spu.h>
extern int edgeZlibFetchAndInflateRawData
(
    uint32_t eaUncompOutput,
    uint32_t expectedUncompSize,
    uint32_t eaCompressed,
    uint32_t compressedSize,
    uint32_t dmaTag,
    unsigned char* pLsInputTempBuffer,
    uint32_t inputTempBuffSize,
    unsigned char* pLsOutputTempBuffer,
    uint32_t outputTempBuffSize
);
```

Arguments

<i>eaUncomOutput</i>	Effective Address of where the uncompressed data will be written to.
<i>expectedUncompSize</i>	The expected output size of the uncompressed data.
<i>eaCompressed</i>	The output buffer at <i>eaUncompOutput</i> must be at least this large.
<i>compressedSize</i>	Effective Address of where the compressed data will be read from.
<i>dmaTag</i>	Size of the compressed data to be read.
<i>pLsInputTempBuffer</i>	The DMA tag to be used for DMAs performed in this function.
<i>inputTempBuffSize</i>	Pointer to the temporary buffer in Local Store to be used for reading input data into.
<i>pLsOutputTempBuffer</i>	The size of this buffer is specified by <i>inputTempBuffSize</i> .
<i>outputTempBuffSize</i>	Size of the temporary buffer point to by <i>pLsInputTempBuffer</i> .
	Pointer to the temporary buffer in Local Store to be used for computing the output data into and sending out from.
	The size of this buffer is specified by <i>outputTempBuffSize</i> .
	The size of the temporary buffer pointed to by <i>outputTempBuffSize</i> .

Return Values

Zero on success; nonzero on failure.

Description

This function decompresses a buffer of compressed data from main memory. This function fetches data into LS and sends data out of LS as necessary.

Notes

This function requires two temporary buffers in LS. These can be specified as parameters from the calling function (*pLsInputTempBuffer* and *pLsOutputTempBuffer*).

This function will assert if the *expectedUncompSize* disagrees with the actual uncompressed size.

This function expects a pointer to the raw compressed data without any header.

If any data error is encountered, this function will not assert (unless a conditional define is changed) and instead returns an error value to its caller.