

# **PlayStation®Edge Zlib**

## **ライブラリ リファレンス**

© 2010 Sony Computer Entertainment Inc.  
All Rights Reserved.  
SCE Confidential

# 目次

はじめに .....	3
このドキュメントについて .....	4
<b>PPU/SPU共有ランタイムのデータ型 .....</b>	<b>5</b>
EdgeZlibDeflateQueueElement .....	6
EdgeZlibInflateQueueElement .....	7
EdgeZlibDeflateQHandle .....	9
EdgeZlibInflateQHandle .....	10
EdgeZlibDeflateTaskProcessing .....	11
EdgeZlibInflateTaskProcessing .....	12
EdgeZlibDeflateTaskProcessingStored .....	13
<b>PPU/SPU共有関数 .....</b>	<b>14</b>
edgeZlibAddDeflateQueueElement .....	15
edgeZlibTryAddDeflateQueueElement .....	17
edgeZlibAddInflateQueueElement .....	19
edgeZlibTryAddInflateQueueElement .....	21
edgeZlibAddInflateQueueElementPartialCopyOut .....	23
edgeZlibTryAddInflateQueueElementPartialCopyOut .....	25
<b>PPU関数 .....</b>	<b>28</b>
edgeZlibGetDeflateQueueSize .....	29
edgeZlibGetInflateQueueSize .....	30
edgeZlibCreateDeflateQueue .....	31
edgeZlibCreateInflateQueue .....	32
edgeZlibShutdownDeflateQueue .....	34
edgeZlibShutdownInflateQueue .....	35
edgeZlibGetDeflateTaskContextSaveSize .....	36
edgeZlibGetInflateTaskContextSaveSize .....	37
edgeZlibCreateDeflateTask .....	38
edgeZlibCreateInflateTask .....	40
<b>SPU関数 .....</b>	<b>42</b>
edgeZlibDeflateRawData .....	43
edgeZlibInflateRawData .....	44
edgeZlibFetchAndDeflateRawData .....	45
edgeZlibFetchAndInflateRawData .....	47

# はじめに

# このドキュメントについて

## 目的

このドキュメントは、Edge ライブラリの Zlib コンポーネントの API リファレンスです。このコンポーネントは、データの伸長や移動を、SPU を通して効率的に行うために使われます。

## 対象読者、および前提条件

このドキュメントは、PlayStation®3 用の高性能アプリケーションを書こうとしている PlayStation®3 デイベロッパのため書かれたものです。このドキュメントでは、そのようなデイベロッパが、以下の知識を持っていることを前提としています。

- C および C++
- PlayStation®3 のハードウェア
- SCE の標準ライブラリ関数

## 関連ドキュメント

このリファレンスと以下のドキュメントを併用することにより、Edge の使用法やリファレンスについての完全な情報を得ることができます。

- PlayStation® Edge ライブラリ概要
- PlayStation® Edge ジオメトリライブラリ キックスタート
- PlayStation® Edge ジオメトリライブラリ リファレンス
- PlayStation® Edge オフラインツール用ジオメトリライブラリ リファレンス
- PlayStation® Edge アニメーションライブラリ リファレンス
- PlayStation® Edge オフラインツール用アニメーションライブラリ リファレンス
- PlayStation® Edge LZMA ライブラリ リファレンス
- PlayStation® Edge LZO ライブラリ リファレンス
- PlayStation® Edge DXT ライブラリ リファレンス
- PlayStation® Edge Post ライブラリ リファレンス

## 表記法

このドキュメントでは、以下のような表記法を使います。

記法	意味
等幅フォント	プログラミングコードおよびリテラル（処理命令、レジスタ名、データ型、イベント、ファイル名など）を表します。また、関数、構造体、マクロなどの名前を表すこともあります。
等幅フォント+太字	構造体や関数の定義の中でのみ、構造体や関数の名前を示します。
等幅フォント+斜体	引数、パラメータ、変数を表します。
<a href="#">青色+下線のテキスト</a>	ハイパーリンクを表します（青色に表示されるのはカラープリンタやオンラインのみ）。

## PPU/SPU共有ランタイムのデータ型

# EdgeZlibDeflateQueueElement

圧縮対象のデータセグメントを圧縮タスクに知らせるキューの要素

## 定義

```
#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)));
```

## メンバ

<i>m_eaInputUncompressedData</i>	入力される圧縮前のデータの実効アドレス
<i>m_eaOutputCompressedData</i>	圧縮されたデータが配置される実効アドレス
<i>m_uncompressedSize</i>	圧縮前のデータのサイズ
<i>m_maxCompressedOutputSize</i>	圧縮されたデータ用の出力バッファの最大サイズ
<i>m_eaOutputCompressedSize</i>	SPU が出力データのサイズを返す実効アドレス。データが圧縮された場合には、最上位ビットがセットされる
<i>m_eaWorkToDoCounter</i>	圧縮完了時にアトミックにデクリメントされるカウンタの実効アドレス。 NULL であってもかまわない。
	最下位 2 ビットは、EdgeZlibDeflateTaskProcessing型のフラグになっている
<i>m_eaEventFlag</i>	イベントフラグの実効アドレス。
	NULL であってもかまわない
<i>m_eventFlagBits</i>	イベントフラグは、伸張が完了後にこの値に設定される
<i>m_compressionLevel</i>	圧縮レベル (1-速度最高、9-サイズ最小)
<i>m_pad8</i>	無視される

## 解説

この構造体には、特定のデータセグメントに関する情報が含まれています。各エントリが、このセグメントを圧縮するための圧縮タスクを起動します。

## 注意

伸張前および伸張後のデータのアドレスは、任意のアラインメントを持つことができます。

## 関連項目

[edgeZlibAddDeflateQueueElement](#)、[edgeZlibCreateDeflateQueue](#)

# EdgeZlibInflateQueueElement

伸張・移動対象のデータセグメントを伸長タスクに知らせるキューの要素

## 定 義

```
#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)));
```

## メ ン バ

<code>m_eaCompressed</code>	伸張前データの実効アドレス
<code>m_eaUncompressed</code>	伸張後のデータが配置される実効アドレス
<code>m_compressedSize</code>	伸張前データのサイズ
<code>m_outputUncompPartialBuffSize</code>	DMA 転送される伸張後のデータのサイズ
<code>m_eaWorkToDoCounter</code>	伸長完了時にアトミックにデクリメントされるカウンタの実効アドレス。 NULL でもかまわない。
	最下位ビットは、セグメントが保存時に圧縮されたか(1)されていないか(0)を示すフラグ
<code>m_eaEventFlag</code>	イベントフラグの実効アドレス。 NULL であってもかまわない
<code>m_eventFlagBits</code>	イベントフラグは、伸長が完了後、この値に設定される
<code>m_outputUncompSkipBeginSize</code>	伸張後の出力のうち最初の N バイトを出力しない
<code>m_outputUncompSkipEndSize</code>	伸張後の出力のうち最後の N バイトを出力しない
<code>m_pad16</code>	無視される

## 解 説

この構造体には、特定のデータセグメントに関する情報が含まれています。この構造体の各要素に応じて、伸長タスクはメモリの伸張や移動を行います。

## 注 意

伸張前および伸張後のデータのアドレスは、任意のアラインメントを持つことができます。

`m_outputUncompSkipBeginSize`、`m_outputUncompPartialBuffSize`、`m_outputUncompSkipEndSize` の 3 つを足すと、伸張後の圧縮データの予想サイズになります。この値が誤っている場合、SPU コードはアサーションを生成します。

## 関 連 項 目

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



---

# EdgeZlibDeflateQHandle

---

## 圧縮キューのハンドル

### 定 義 ( P P U の 場 合 )

---

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

### 定 義 ( S P U の 場 合 )

---

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

### 解 説

---

この型は、メインメモリ上の圧縮キューを指すハンドルです。

### 関 連 項 目

---

[EdgeZlibDeflateQueueElement](#)、[edgeZlibCreateDeflateQueue](#)

---

# EdgeZlibInflateQHandle

---

## 伸長キューのハンドル

### 定義（ P P U の 場 合 ）

---

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

### 定義（ S P U の 場 合 ）

---

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

### 解 説

---

この型は、メインメモリ上の伸長キューを指すハンドルです。

### 関 連 項 目

---

[EdgeZlibInflateQueueElement](#)、[edgeZlibCreateInflateQueue](#)

---

# EdgeZlibDeflateTaskProcessing

---

圧縮後のセグメントを格納するか、それとも圧縮後と圧縮前のデータのうち小さいほうを格納するかを指定する

## 定 義

---

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

## メ ン バ

---

<i>kEdgeZlibDeflateTask_DeflateStoreCompressed</i>	常に圧縮されたデータを格納する
<i>kEdgeZlibDeflateTask_DeflateStoreSmallest</i>	圧縮後と圧縮前のデータのうち小さい方を格納する
<i>kEdgeZlibDeflateTask_DeflateStoreCompressedWithHeader</i>	常に、圧縮後のデータを格納。その際、zlib の 2 バイトのヘッダと 4 バイトのフッタを付加する

## 解 説

---

この型は、圧縮後のデータに zlib の 2 バイトのヘッダと 4 バイトのフッタを付加するかどうか、および圧縮によってサイズが減らない場合には、圧縮後データと圧縮前のデータのうち小さい方を（ヘッダやフッタなしで）格納するかどうかを宣言するために使われます。

## 関 連 項 目

---

[edgeZlibAddDeflateQueueElement](#)

---

# EdgeZlibInflateTaskProcessing

---

セグメントをコピーするか伸張するかを指定する

## 定 義

---

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

## メ ン バ

---

<i>kEdgeZlibInflateTask_Memcpy</i>	特定の場所から別の場所にメモリをコピーする
<i>kEdgeZlibInflateTask_Inflate</i>	特定の場所から別の場所にメモリを伸張する

## 解 説

---

この型は、伸長タスクがセグメントをコピーするか伸張するかを宣言するために使われます。

## 関 連 項 目

---

[edgeZlibAddInflateQueueElement](#)、[edgeZlibAddInflateQueueElementPartialCopyOut](#)

---

# EdgeZlibDeflateTaskProcessingStored

---

圧縮タスクが、出力データが圧縮されて格納されているかどうか指定するために使用される

## 定 義

---

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

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

## メ ン バ

---

<i>kEdgeZlibDeflateTask_UncompressedWasStored</i>	格納されたのは圧縮前のデータである
<i>kEdgeZlibDeflateTask_CompressedWasStored</i>	格納されたのは圧縮後のデータである (オプションの zlib ヘッダ、フッタは ある場合もない場合もある)

## 解 説

---

SPU が出力サイズを書き込むとき、格納されたデータが圧縮されているかいないかを宣言するために、最上位ビットが使われます。圧縮データには、ユーザの要求に応じて、オプションで、zlib の 2 バイトのヘッダと 4 バイトのフッタが付加されることもあります。

## 関 連 項 目

---

[edgeZlibAddDeflateQueueElement](#)

## PPU/SPU共有関数

# edgeZlibAddDeflateQueueElement

圧縮キューに新しい作業を追加する

## 定義 ( PPU の 場 合 )

```
#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
)
```

## 定義 ( SPU の 場 合 )

```
#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
)
```

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

<i>handle</i>	要素がプッシュされる圧縮キューのハンドル
<i>eaInputUncompressedData</i>	圧縮される入力データの実効アドレス。 これは、ヘッダのない生データへのポインタである必要がある
<i>uncompressedSize</i>	圧縮前の入力データのサイズ

<code>eaOutputCompressedData</code>	圧縮後のデータ用の出力バッファの実効アドレス
<code>maxCompressedOutputSize</code>	圧縮データバッファに利用できる最大領域
<code>eaOutputCompressedSize</code>	出力圧縮サイズが書き込まれる <code>uint32_t</code> の実効アドレス。 書き込まれたカウンタの最上位ビットは、データが圧縮されて格納されたか、それとも圧縮前の元のデータが格納されることが選択されたかを示す
<code>eaWorkToDoCounter</code>	この要素の伸張後にアトミックにデクリメントされるメインメモリ上のカウンタ。データ圧縮時、SPU がエラーを持つ場合には、エラーフラグとして上位ビットがセットされる。
<code>eaEventFlag</code>	NULL であってもかまわない 設定するイベントフラグ。
<code>eventFlagBits</code>	NULL であってもかまわない
<code>level</code>	圧縮完了後にイベントフラグはこの値に設定される。 圧縮レベルを 0～9 の間で選択する。 0 - 圧縮なし 1 - 速度最高 9 - サイズ最小
<code>processing</code>	圧縮データにオプションの zlib の 2 バイトのヘッダおよび 4 バイトのフッタを付加するかどうか、および圧縮後と圧縮前のデータのうちより小さい方を格納するかどうかを選択する
<code>dmaTag</code>	(SPU のみ) この関数内で実行される DMA で使われる DMA タグ

## 返 り 値

なし。

## 解 説

圧縮キューに新しい作業を追加します。このキューに追加された作業は、次の機会に、圧縮タスクに渡されます。

## 注 意

この関数は、キューが一杯（つまり、`maxNumQueueEntries` に到達した場合）のときに、キューに別の要素がプッシュされた場合には、開き領域ができるまでブロックします。

作業数カウンタ（`eaWorkToDoCounter`）を待機中のセグメント数に初期化すれば、複数セグメントに対する処理が完了したかどうかを追跡するために利用できます。このカウンタは、各セグメントに対する処理が完了するたびに 1 つデクリメントされるので、このカウンタの値がゼロになれば、全セグメントに対する処理が完了しているはずです。この時点で、指定されたイベントフラグも設定されます。データ圧縮時 SPU にエラーがある場合には、そのエラーがカウンタの上位ビットを設定することに注意してください。

`eeaWorkToDoCounter` は NULL であるにもかかわらず、`eaEventFlag` が有効である場合、イベントフラグは、このセグメントに対する処理が完了した後で設定されます。

## 関 連 項 目

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



# edgeZlibTryAddDeflateQueueElement

圧縮キューに新しい作業を追加することを試みる

## 定義 ( PPU の 場 合 )

```
#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
)
```

## 定義 ( SPU の 場 合 )

```
#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
)
```

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

<i>handle</i>	要素がプッシュされる圧縮キューのハンドル
<i>eaInputUncompressedData</i>	圧縮される入力データの実効アドレス。 これは、ヘッダのない生データへのポインタである必要がある
<i>uncompressedSize</i>	圧縮前の入力データのサイズ

<code>eaOutputCompressedData</code>	圧縮後のデータ用の出力バッファの実効アドレス
<code>maxCompressedOutputSize</code>	圧縮データに利用できる最大領域
<code>eaOutputCompressedSize</code>	出力圧縮サイズが書き込まれる <code>uint32_t</code> の実効アドレス。 書き込まれたカウンタの最上位ビットは、データが圧縮されて格納されたか、それとも圧縮されていない元のデータを保存することが選択されたかを示している
<code>eaWorkToDoCounter</code>	この要素の伸張後にアトミックにデクリメントされるメインメモリ上のカウンタ。データ圧縮時に SPU にエラーがあった場合には、エラーフラグとして上位ビットがセットされる。
<code>eaEventFlag</code>	NULL であってもかまわない 設定するイベントフラグ。
<code>eventFlagBits</code>	NULL であってもかまわない
<code>level</code>	圧縮完了後にイベントフラグはこの値に設定される 圧縮レベルを 0～9 の間で選択する。 0 - 圧縮なし 1 - 速度最高 9 - サイズ最小
<code>processing</code>	圧縮データにオプションの zlib の 2 バイトのヘッダおよび 4 バイトのフッタを付加するかどうか、および圧縮後と圧縮前のデータでより小さい方を格納するかどうかを選択します
<code>dmaTag</code>	(SPU のみ) この関数内で実行される DMA で使われる DMA タグ

## 返 り 値

圧縮キューに作業が追加された場合には `true`。追加できなかった場合には `false`。

## 解 説

圧縮キューに新しい作業を追加することを試みます。追加できた場合、このキューに追加された作業は、次の機会に、圧縮タスクに渡されます。

## 注 意

この関数は、キューが一杯（つまり、`maxNumQueueEntries` に到達した場合）でキューに別の要素がプッシュされた場合には、ただちに `false` を返します。

作業数カウンタ（`eaWorkToDoCounter`）を待機中のセグメント数に初期化すれば、複数セグメントに対する処理が完了したかどうかを追跡するために利用できます。このカウンタは、各セグメントに対する処理が完了するたびに 1 つデクリメントされるので、このカウンタの値がゼロになれば、全セグメントに対する処理が完了しているはずです。この時点で、指定されたイベントフラグも設定されます。データ圧縮時に SPU でエラーが発生した場合には、そのエラーがカウンタの上位ビットを設定することに注意してください。

`eaWorkToDoCounter` は NULL であるにもかかわらず、`eaEventFlag` が有効である場合、イベントフラグは、このセグメントに対する処理が完了した後で設定されます。

## 関 連 項 目

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

# edgeZlibAddInflateQueueElement

伸長キューに新しい作業を追加する

## 定義 ( P P U の 場 合 )

```
#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
)
```

## 定義 ( S P U の 場 合 )

```
#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
)
```

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフ。

## 引 数

<i>handle</i>	要素がプッシュされる伸長キューのハンドル
<i>eaInputCompressedData</i>	伸張される入力データの実効アドレス。 これは、ヘッダのない生データへのポインタである必要がある
<i>compressedSize</i>	伸張前の入力データのサイズ
<i>eaOutputUncompressed</i>	伸張後のデータ用の出力バッファの実効アドレス
<i>expectedUncompressedSize</i>	伸張後のデータの予想サイズ。 この値が誤っていると、SPU はアサーションを生成する

<code>eaWorkToDoCounter</code>	この要素の伸張後にアトミックにデクリメントされるメインメモリ上のカウンタ。
<code>eaEventFlag</code>	NULL でもかまわない 設定するイベントフラグ。
<code>eventFlagBits processing</code>	NULL でもかまわない イベントフラグは、伸長が完了後にこの値に設定される タスクがデータに対して行う処理の種類を選択する。 伸長を実行することも、単に <code>eaInputCompressedData</code> から <code>eaOutputUncompressedData</code> まで生データを移動することも できる
<code>dmaTag</code>	(SPU のみ) この関数内で実行される DMA で使われる DMA タグ

## 返 り 値

なし

## 解 説

伸長キューに新しい作業を追加します。このキューに追加された作業は、次の機会に、伸長タスクに渡されます。

## 注 意

この関数は、キューが一杯（つまり、`maxNumQueueEntries` に到達した場合）のときに、キューに別の要素がプッシュされた場合には、開き領域ができるまでブロックします。

作業数カウンタ（`eaWorkToDoCounter`）を待機中のセグメント数に初期化すれば、複数セグメントに対する処理が完了したかどうかを追跡するために利用できます。このカウンタは、各セグメントに対する処理が完了するたびにデクリメントされるので、このカウンタの値がゼロになれば、全セグメントに対する処理が完了したことを意味します。この時点で、指定されたイベントフラグが設定されます。データ圧縮時に SPU でエラーが発生した場合には、そのエラーがカウンタの上位ビットに設定されることに注意してください。

`eaWorkToDoCounter` が NULL であるにもかかわらず、`eaEventFlag` が有効である場合、イベントフラグは、このセグメントに対する処理が完了した後で設定されます。

保存データが圧縮されておらず、単に移動先に移動することだけが必要な場合にも、伸長タスクにメモリコピーだけを実行することを示す値を渡せば、この処理を伸長タスクを利用して行うことが可能です。この関数に渡すのは、ヘッダのない圧縮された生データへのポインタである必要があります。

## 関 連 項 目

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

# edgeZlibTryAddInflateQueueElement

伸長キューに新しい作業を追加することを試みる

## 定義 ( P P U の 場 合 )

```
#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
)
```

## 定義 ( S P U の 場 合 )

```
#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
)
```

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

<i>handle</i>	要素がプッシュされる伸長キューのハンドル
<i>eaInputCompressedData</i>	伸張される入力データの実効アドレス。 これは、ヘッダのない生データへのポインタである必要がある
<i>compressedSize</i>	圧縮された入力データのサイズ
<i>eaOutputUncompressed</i>	圧縮されていないデータ用の出力バッファの実効アドレス
<i>expectedUncompressedSize</i>	圧縮されていないデータの予想サイズ。 この値が誤っていると、SPU はアサーションを生成する

<i>eaWorkToDoCounter</i>	この要素の伸張後にアトミックにデクリメントされるメインメモリ上のカウンタ。データ圧縮時に SPU でエラーが発生した場合には、カウンタの上位ビットがエラーフラグとして設定される。
<i>eaEventFlag</i>	NULL であってもかまわない 設定するイベントフラグ。
<i>eventFlagBits</i>	NULL であってもかまわない イベントフラグに設定する値
<i>processing</i>	タスクがデータに対して行う処理の種類を選択する。 伸長を実行することも、単に <i>eaInputCompressedData</i> から <i>eaOutputUncompressedData</i> まで生データを移動することもできる
<i>dmaTag</i>	この関数内で実行される DMA で使われる DMA タグ (SPU のみ)

## 返 り 値

キューに作業を追加することに成功した場合には true。追加に失敗した場合には false。

## 解 説

伸長キューに新しい作業を追加を試みます。このキューに追加された作業は、次の機会に、伸長タスクに渡されます。

## 注 意

この関数は、作業が追加された場合に、true を返します。この関数は、キューが一杯のときに（つまり、*maxNumQueueEntries* に到達した場合）、キューに別の要素がプッシュされた場合には、false を返します。

作業数カウンタ (*eaWorkToDoCounter*) を待機中のセグメント数に初期化すれば、複数セグメントに対する処理が完了したかどうかを追跡するために利用できます。このカウンタは、各セグメントに対する処理が完了するたびにデクリメントされるので、このカウンタの値がゼロになれば、全セグメントに対する処理が完了しているはずです。この時点で、指定されたイベントフラグも設定されます。SPU が圧縮データでエラーを持つ場合、それがカウンタの高位ビットを設定する点に注意します。

*eaWorkToDoCounter* が NULL であるにもかかわらず、*eaEventFlag* が有効である場合、イベントフラグは、このセグメントに対する処理が完了した後で設定されます。

保存データが圧縮されずに格納され、単に移動先に移動することだけが必要な場合にも、伸長タスクに処理用の適切な値を渡せば、伸長タスクを利用してこの処理を行うことが可能です。

この関数に渡すのは、ヘッダのない生データへのポインタである必要があります。

## 関 連 項 目

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

# edgeZlibAddInflateQueueElementPartialCopyOut

伸長キューに新しい作業を追加する

## 定義（ P P U の 場 合 ）

```
#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
)
```

## 定義（ S P U の 場 合 ）

```
#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
)
```

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフ。

## 引 数

<i>handle</i>	要素がプッシュされる伸長キューのハンドル
<i>eaInputCompressedData</i>	伸張される入力データの実効アドレス。 これは、ヘッダのない生データへのポインタである必要がある 伸張前の入力データのサイズ
<i>compressedSize</i>	

<code>eaOutputUncompPartialBuff</code>	伸張後のデータ用の出力バッファの実効アドレス
<code>outputUncompSkipBeginSize</code>	伸張後の出力のうち最初の N バイトを出力しない
<code>outputUncompPartialBuffSize</code>	DMA 転送される伸張後のデータのサイズ
<code>outputUncompSkipEndSize</code>	伸張後の出力のうち最後の N バイトを出力しない
<code>eaWorkToDoCounter</code>	この要素の伸張後にアトミックにデクリメントされるメインメモリ上のカウンタ。
	NULL でもかわない
<code>eaEventFlag</code>	設定するイベントフラグ。
	NULL でもかわない
<code>eventFlagBits</code>	イベントフラグに設定する値
<code>processing</code>	タスクがデータに対して行う処理の種類を選択する。
	伸長を実行することも、単に <code>eaInputCompressedData</code> から <code>eaOutputUncompressedData</code> まで生データを移動することもできる
<code>dmaTag</code>	この関数内で実行される DMA で使われる DMA タグ (SPU のみ)

## 返 り 値

なし

## 解 説

伸長キューに新しい作業を追加します。このキューに追加された作業は、次の機会に、伸長タスクに渡されます。

## 注 意

この関数は、キューが一杯のときに（つまり、`maxNumQueueEntries` に到達した場合）、キューに別の要素がプッシュされた場合には、開き領域ができるまでブロックします。

作業数カウンタ (`eaWorkToDoCounter`) を待機中のセグメント数に初期化すれば、複数セグメントに対する処理が完了したかどうかを追跡するために利用できます。このカウンタは、各セグメントに対する処理が完了するたびにデクリメントされるので、このカウンタの値がゼロになれば、全セグメントに対する処理が完了したことを意味します。この時点で、指定されたイベントフラグが設定されます。`eaWorkToDoCounter` が NULL であるにもかかわらず、`eaEventFlag` が有効である場合、イベントフラグは、このセグメントに対する処理が完了した後で設定されます。SPU が圧縮データでエラーを持つ場合、それがカウンタの高位ビットを設定する点に注意します。

保存データが圧縮されておらず、単に移動先に移動することだけが必要な場合にも、伸長タスクに「処理」用の適切な値を渡せば、伸長タスクを利用してこの処理を行うことが可能です。

この関数に渡すのは、ヘッダのない生データへのポインタである必要があります。

この関数は、[edgeZlibAddInflateQueueElement](#) に非常によく似ています。追加された `skipOutputBeginSize` および `skipOutputEndSize` という 2 つのパラメータは、セグメントの伸長は行いたい、書き込む必要があるのは出力の一部だけである場合のためのものです。

`outputUncompSkipBeginSize`、`outputUncompPartialBuffSize`、`outputUncompSkipEndSize` を足すと、圧縮データの伸張後の予想サイズになります。この値が誤っていると、SPU はアサーションを生成します

## 関 連 項 目

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



# edgeZlibTryAddInflateQueueElementPartialCopyOut

伸長キューに新しい作業を追加することを試みる

## 定義（ P P U の 場 合 ）

```
#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
)
```

## 定義（ S P U の 場 合 ）

```
#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
)
```

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

<i>handle</i>	要素がプッシュされる伸長キューのハンドル
<i>eaInputCompressedData</i>	伸張される入力データの実効アドレス。 これは、ヘッダのない生データへのポインタである必要がある
<i>compressedSize</i>	圧縮された入力データのサイズ

<code>eaOutputUncompPartialBuff</code>	圧縮されていないデータ用の出力バッファの実効アドレス
<code>outputUncompSkipBeginSize</code>	圧縮されていない出力のうち最初の N バイトを出力しない
<code>outputUncompPartialBuffSize</code>	DMA 転送される圧縮されていないデータのサイズ
<code>outputUncompSkipEndSize</code>	圧縮されていない出力のうち最後の N バイトを出力しない
<code>eaWorkToDoCounter</code>	この要素の伸張後にアトミックにデクリメントされるメインメモリ上のカウンタ。データ圧縮時に SPU でエラーが発生した場合には、そのエラーがエラーフラグとしてカウンタの上位ビットに設定される。
<code>eaEventFlag</code>	NULL であってもかまわない 設定するイベントフラグ。
<code>eventFlagBits</code>	NULL であってもかまわない イベントフラグに設定する値
<code>processing</code>	タスクがデータに対して行う処理の種類を選択する。 伸長を実行することも、単に <code>eaInputCompressedData</code> から <code>eaOutputUncompressedData</code> まで生データを移動することもできる
<code>dmaTag</code>	この関数内で実行される DMA で使われる DMA タグ (SPU のみ)

## 返 り 値

キューに作業を追加することに成功した場合には true。追加に失敗した場合には false。

## 解 説

伸長キューに新しい作業を追加することを試みます。このキューに追加された作業は、次の機会に、伸長タスクに渡されます。

## 注 意

この関数は、作業が追加された場合に、true を返します。この関数は、キューが一杯のときに（つまり、`maxNumQueueEntries` に到達した場合）、キューに別の要素がプッシュされた場合には、false を返します。

作業数カウンタ (`eaWorkToDoCounter`) を待機中のセグメント数に初期化すれば、複数セグメントに対する処理が完了したかどうかを追跡するために利用できます。このカウンタは、各セグメントに対する処理が完了するたびにデクリメントされるので、このカウンタの値がゼロになれば、全セグメントに対する処理が完了しているはずです。この時点で、指定されたイベントフラグも設定されます。SPU が圧縮データでエラーを持つ場合、それがカウンタの高位ビットを設定する点に注意します。

`eaWorkToDoCounter` は NULL であるにもかかわらず、`eaEventFlag` が有効である場合、イベントフラグは、このセグメントに対する処理が完了した後で設定されます。

保存データが圧縮されずに格納され、単に移動先に移動することだけが必要な場合にも、伸長タスクに処理用の適切な値を渡せば、伸長タスクを利用してこの処理を行うことが可能です。

この関数に渡すのは、ヘッダのない生データへのポインタである必要があります。

この関数は、[edgeZlibAddInflateQueueElement](#) に非常によく似ています。追加された `skipOutputBeginSize` および `skipOutputEndSize` という 2 つのパラメータは、セグメントの伸長は行いたい、書き込む必要があるのは出力の一部だけである場合のためのものです。

`outputUncompSkipBeginSize`、`outputUncompPartialBuffSize`、`outputUncompSkipEndSize` を足すと、圧縮データの伸張後の予想サイズが得られます。この値が誤っていると、SPU はアサーションを生成します

## 関 連 項 目

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

# PPU関数

---

# edgeZlibGetDeflateQueueSize

---

圧縮キューに必要なバッファサイズを返す

## 定 義

---

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

## 呼 出 条 件

---

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

---

*maxNumQueueEntries* キューが同時に保持する要素の最大数（最大：32767）

## 返 り 値

---

必要なバッファサイズを返します。

## 解 説

---

圧縮キューが指定された要素数を保持するために必要なバッファサイズを返します。

## 注 意

---

割り当てられたバッファは、128 バイトにアラインメントされている必要があります。

## 関 連 項 目

---

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

---

# edgeZlibGetInflateQueueSize

---

伸長キューに必要なバッファサイズを返す

## 定 義

---

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

## 呼 出 条 件

---

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフ。

## 引 数

---

*maxNumQueueEntries* キューが同時に保持する要素の最大数（最大：32767）

## 返 り 値

---

必要なバッファサイズを返します。

## 解 説

---

伸長キューが指定された要素数を保持するために必要なバッファサイズを返します。

## 注 意

---

割り当てられたバッファは、128 バイトにアラインメントされている必要があります。

## 関 連 項 目

---

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

# edgeZlibCreateDeflateQueue

## 圧縮キューを作成する

### 定 義

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

### 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

### 引 数

<i>pSpurs</i>	この圧縮キューが関連付けられる SPURS インスタンスへのポインタ
<i>maxNumQueueEntries</i>	このキューが同時に保持する要素の最大数（最大：32767）
<i>pBuffer</i>	この圧縮キュー用に使われるメインメモリ上のバッファへのポインタ。 128 バイト境界にアラインメントされている必要がある
<i>bufferSize</i>	メインメモリ上に提供されたバッファのサイズ。 指定された数のキュー要素のために必要なバッファのサイズ は、 <a href="#">edgeZlibGetDeflateQueueSize()</a> を呼び出すことによって取得できる

### 返 り 値

作成された圧縮キューの [EdgeZlibDeflateQHandle](#)。

### 解 説

圧縮キューを作成します。圧縮キューは、圧縮タスクに処理させるためにキューに入れられた作業（圧縮するセグメント）のリストを保持します。

### 注 意

圧縮キューは FIFO で、キューが一杯のときに作業がプッシュされたような場合には、必要に応じてストールするので、キューの長さは、実際に必要な最大要素数より短くてもかまいません。

### 関 連 項 目

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

# edgeZlibCreateInflateQueue

## 伸長キューを作成する

### 定 義

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

### 呼 出 条 件

割り込みハンドラから呼び出し可能。スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。マルチスレッドセーフ。

### 引 数

<i>pSpurs</i>	この伸長キューが関連付けられる SPURS インスタンスへのポインタ
<i>maxNumQueueEntries</i>	このキューが同時に保持する要素の最大数（最大：32767）
<i>pBuffer</i>	この伸長キュー用に使われるメインメモリ上のバッファへのポインタ。 128 バイト境界にアラインメントされている必要がある
<i>bufferSize</i>	メインメモリ上に提供されたバッファのサイズ。 指定された数のキュー要素のために必要なバッファのサイズ は、 <a href="#">edgeZlibGetInflateQueueSize()</a> を呼び出すことによって取得できる

### 返 り 値

作成された伸長キューの [EdgeZlibInflateQHandle](#)。

### 解 説

伸長キューを作成します。伸張キューは、伸長タスクに処理させるためにキューに入れられた作業（伸張・移動するセグメント）のリストを保持します。

### 注 意

伸長キューは FIFO で、キューが一杯のときに作業がプッシュされたような場合には、必要に応じてストールするので、キューの長さは、実際に必要な最大要素数より短くてもかまいません。

### 関 連 項 目

[EdgeZlibInflateQueueElement](#)、[edgeZlibAddInflateQueueElement](#)、[edgeZlibAddInflateQueueElementPartialCopyOut](#)、[edgeZlibGetInflateQueueSize](#)、[edgeZlibShutdownInflateQueue](#)、[edgeZlibCreateInflateTask](#)





---

# edgeZlibShutdownDeflateQueue

---

圧縮キューをシャットダウンする

## 定 義

---

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

## 呼 出 条 件

---

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

---

*handle*      シャットダウンする圧縮キューのハンドル

## 返 り 値

---

なし。

## 解 説

---

圧縮キューをシャットダウンします。

## 関 連 項 目

---

[edgeZlibCreateDeflateQueue](#)

---

# edgeZlibShutdownInflateQueue

---

伸長キューをシャットダウンする

## 定 義

---

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

## 呼 出 条 件

---

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフ。

## 引 数

---

*handle*      シャットダウンする伸長キューのハンドル

## 返 り 値

---

なし

## 解 説

---

伸長キューをシャットダウンします。

## 関 連 項 目

---

[edgeZlibCreateInflateQueue](#)

---

# edgeZlibGetDeflateTaskContextSaveSize

---

圧縮タスクがコンテキストを格納するために必要なバッファサイズを返す

## 定 義

---

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

## 呼 出 条 件

---

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能。（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

---

(なし)

## 返 り 値

---

単一の圧縮タスクのコンテキストデータを格納するために必要なバッファのサイズ。

## 解 説

---

この関数は、単一の圧縮タスクのコンテキストを格納するために、ライブラリが必要とするバッファサイズを返します。

## 注 意

---

割り当てられたバッファは、16 バイトにアラインメントされている必要があります。

## 関 連 項 目

---

[edgeZlibCreateDeflateTask](#)

---

# edgeZlibGetInflateTaskContextSaveSize

---

伸長タスクがコンテキスト情報を格納するために必要なバッファサイズを返す

## 定 義

---

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

## 呼 出 条 件

---

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフ。

## 引 数

---

(なし)

## 返 り 値

---

単一の伸長タスクのコンテキストデータを格納するために必要なバッファのサイズ。

## 解 説

---

この関数は、単一の伸長タスクのコンテキストを格納するために、ライブラリが必要とするバッファサイズを返します。

## 注 意

---

割り当てられたバッファは、16 バイトにアラインメントされている必要があります。

## 関 連 項 目

---

[edgeZlibCreateInflateTask](#)

# edgeZlibCreateDeflateTask

単一の SPU 上で圧縮を実行する SPURS タスクを作成する

## 定 義

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

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能。（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフである。

## 引 数

<i>pTaskSet</i>	この圧縮タスクが関連付けられる SPURS タスクセットへのポインタ
<i>pTaskContext</i>	タスクがコンテキストを格納するために使う、メインメモリ上のバッファへのポインタ。 このバッファに必要なサイズは、 <a href="#">edgeZlibGetDeflateTaskContextSaveSize</a> を呼び出すことによって取得できる
<i>handle</i>	このタスクが取得される圧縮キューのハンドル

## 返 り 値

作成されたタスクの CellSpursTaskId。

## 解 説

このタスクは、圧縮キューに作業が存在する場合には、その作業を実行します。  
実行すべき作業が存在しない場合、タスクは、*pTaskContext* によって指定される場所にコンテキストを格納して、スリープします。

## 注 意

圧縮タスクは、実行したい SPU ごとに 1 つずつ作成します。したがって、6 つの SPU 上で圧縮を並列実行したい（かつ、SPURS インスタンスの中に 6 つの SPU がある）場合には、同じタスクセットの中にあり、同じ圧縮キューに対して処理を行う、6 つの圧縮タスクを作成する必要があります。

*pTaskContext* の指すバッファに必要なサイズ

は、[edgeZlibGetDeflateTaskContextSaveSize\(\)](#) を呼び出すことによって求めることができます。

## 関 連 項 目

[edgeZlibAddDeflateQueueElement](#)、[edgeZlibGetDeflateQueueSize](#)、[edgeZlibCreatedDeflateQueue](#)、[edgeZlibShutdownDeflateQueue](#)、[edgeZlibGetDeflateTaskContextSaveSize](#)

# edgeZlibCreateInflateTask

単一の SPU 上で伸長を実行する SPURS タスクを作成する

## 定 義

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

## 呼 出 条 件

割り込みハンドラから呼び出し可能。  
スレッドから呼び出し可能（割り込み禁止・許可の状態とは無関係）。  
マルチスレッドセーフ。

## 引 数

<i>pTaskSet</i>	この伸長タスクが関連付けられる SPURS タスクセットへのポインタ
<i>pTaskContext</i>	タスクがコンテキストを格納するために使う、メインメモリ上のバッファへのポインタ。 このバッファに必要なサイズは、 <code>edgeZlibGetInflateTaskContextSaveSize</code> を呼び出すことにより取得できる
<i>handle</i>	このタスクを取得する伸長キューのハンドル

## 返 り 値

作成されたタスクの `CellSpursTaskId`。

## 解 説

このタスクは、伸長キューに作業が存在する場合には、その作業を実行します。  
またこのタスクは、実行する作業が存在しない場合には、スリープ状態になります。その際、  
*pTaskContext* によって指定された場所にコンテキストを格納します。

## 注 意

伸長タスクは、実行したい SPU ごとに 1 つずつ作成します。したがって、6 つの SPU 上で伸張を並列実行したい（かつ、SPURS インスタンスの中に 6 つの SPU がある）場合には、同じタスクセットの中にあり、同じ伸長キューに対して処理を行う、6 つの伸長タスクを作成する必要があります。  
*pTaskContext* が指すバッファに必要なサイズは、`edgeZlibGetInflateTaskContextSaveSize` を呼び出すことによって取得できます。



## 関 連 項 目

[edgeZlibAddInflateQueueElement](#)、[edgeZlibAddInflateQueueElementPartialCopyOut](#)、[edgeZlibGetInflateQueueSize](#)、[edgeZlibCreateInflateQueue](#)、[edgeZlibShutdownInflateQueue](#)、[edgeZlibGetInflateTaskContextSaveSize](#)

# SPU関数

# edgeZlibDeflateRawData

圧縮されていないデータのバッファを圧縮する

## 定 義

```
#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,
);
```

## 引 数

<i>pUncompr</i>	圧縮されていないデータが読み込まれるローカルストアのポインタ
<i>uncompSize</i>	圧縮される入力非圧縮データのサイズ
<i>pComprData</i>	圧縮後のデータが書き込まれるローカルストアのポインタ この領域には、ヘッダのない生の圧縮データが書き込まれる
<i>maxCompressedDataSize</i>	圧縮後のデータバッファの最大サイズ
<i>pOutputCompressedSize</i>	圧縮後のデータの出力サイズを返す
<i>Level</i>	圧縮レベルを 0～9 の間で選択する。 0 - 圧縮なし 1 - 速度最高 9 - サイズ最小

## 返 り 値

成功した場合にはゼロ。エラーの場合にはゼロ以外。

## 解 説

この関数は、ローカルストア（LS）中の圧縮されていないデータのバッファを圧縮します。

## 注 意

入力バッファと出力バッファは、ともに LS 中にある必要があります。入力データの提供および出力データの処理は、[edgeZlibDeflateRawData\(\)](#) を呼び出す関数によって行われることが想定されています。この関数は、内部的には DMA を行いません。

---

# edgeZlibInflateRawData

---

## 圧縮データのバッファを伸張する

### 定 義

---

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

### 引 数

---

<i>pUncompr</i>	伸張後のデータが書き込まれるローカルストアのポインタ
<i>expectedUncompSize</i>	伸張後のデータの予想出力サイズ。 <i>pUncompr</i> の指す出力バッファは、最低でもこのサイズ以上である必要がある
<i>pComprData</i>	伸張前のデータが読み込まれるローカルストアのポインタ。 これは、ヘッダのない生データへのポインタである必要がある
<i>comprDataSize</i>	読み込まれる伸張前データのバッファサイズ

### 返 り 値

---

成功した場合はゼロ、エラーの場合にゼロ以外。

### 解 説

---

この関数は、ローカルストア（LS）中の圧縮データのバッファを伸張します。

### 注 意

---

入力バッファと出力バッファは、ともにLSの中にあります。入力データの提供および出力データの処理は、`edgeZlibInflateRawData()` を呼び出す関数によって行われることが想定されています。この関数は、内部的にはDMAを行いません。

この関数は、`expectedUncompSize` が実際の伸張後のサイズと一致しない場合、アサーションを生成します。

この関数に渡すのは、ヘッダのない圧縮された生データへのポインタである必要があります。

この関数は、データエラーが発生した場合、（条件付き定義が変更されない限り）アサーションを発生する代わりに、呼出し元にエラー値を返します。

# edgeZlibFetchAndDeflateRawData

メインメモリ中の未圧縮データのバッファをメインメモリ中の別のバッファに圧縮する

## 定 義

```
#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
);
```

## 引 数

<i>eaOutputCompressedData</i>	圧縮後のデータ用の出力バッファの実効アドレス
<i>maxCompressedOutputSize</i>	圧縮データに利用できる最大領域
<i>eaOutputSize</i>	出力圧縮サイズが書き込まれる uint32_t の実効アドレス。出力サイズの最上位ビットは、データが圧縮されて格納されたか、それとも圧縮されていない元のデータを格納することが選択されたかを示しています
<i>eaInputUncompressedData</i>	圧縮される入力データの実効アドレス。 これは、ヘッダのない生データへのポインタである必要がある
<i>uncompressedSize</i>	圧縮前の入力データのサイズ
<i>dmaTag</i>	この関数内で実行される DMA で使われる DMA タグ
<i>pLsInputTempBuffer</i>	入力データの読み込みに使われるローカルストア中の一時バッファへのポインタ。このバッファのサイズは、 <i>inputTempBuffSize</i> によって指定される
<i>inputTempBuffSize</i>	<i>pLsInputTempBuffer</i> が指す一時バッファのサイズ
<i>pLsOutputTempBuffer</i>	出力データの計算先および送信元として使われる、ローカルストア中の一時バッファへのポインタ。このバッファのサイズは、 <i>outputTempBuffSize</i> によって指定される
<i>outputTempBuffSize</i>	<i>pLsOutputTempBuffer</i> が指す一時バッファのサイズ
<i>level</i>	圧縮レベルを 0~9 の間で選択する。 0 - 圧縮なし 1 - 速度最高 9 - サイズ最小
<i>taskProcessing</i>	圧縮データにオプションの zlib の 2 バイトのヘッダおよび 4 バイトのフッタを付加するかどうか、および圧縮後と圧縮前のデータでより小さい方を保存するかどうかを選択します

## 返 り 値

成功した場合にはゼロ。エラーの場合にはゼロ以外。

## 解 説

---

この関数は、メインメモリ中の圧縮されていないデータのバッファを圧縮します。この関数は、必要に応じて LS からデータの読み込み・送信を行います。

## 注 意

---

この関数は、LS 中に一時バッファを 2 つ必要とします。このバッファは、呼び出し側関数 (`pLsInputTempBuffer`、`pLsOutputTempBuffer`) からパラメータとして指定することができます。

# edgeZlibFetchAndInflateRawData

メインメモリ中の圧縮データのバッファをメインメモリ中の別のバッファに伸張する

## 定 義

```
#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
);
```

## 引 数

<i>eaUncomOutput</i>	圧縮されてないデータが書き込まれる実効アドレス
<i>expectedUncompSize</i>	圧縮されてないデータの予想出力サイズ。 <i>eaUncompOutput</i> にある出力バッファは、最低でもこのサイズ以上ある必要がある
<i>eaCompressed</i>	圧縮後のデータが読み込まれる実効アドレス
<i>compressedSize</i>	読み込まれる伸張前データのサイズ
<i>dmaTag</i>	この関数内で実行される DMA で使われる DMA タグ
<i>pLsInputTempBuffer</i>	入力データの読み込みに使われるローカルストア中の一時バッファへのポインタ。 このバッファのサイズは、 <i>inputTempBuffSize</i> によって指定される
<i>inputTempBuffSize</i>	<i>pLsInputTempBuffer</i> が指す一時バッファのサイズ
<i>pLsOutputTempBuffer</i>	出力データの計算先および送信元として使われるローカルストア中の一時バッファへのポインタ。 このバッファのサイズは、 <i>outputTempBuffSize</i> によって指定される
<i>outputTempBuffSize</i>	<i>outputTempBuffSize</i> が指す一時バッファのサイズ

## 返 り 値

成功した場合にはゼロ。失敗した場合にはゼロ以外の値。

## 解 説

この関数は、メインメモリ中の圧縮データのバッファを伸張します。この関数は、必要に応じて LS からデータの読み込み・送信を行います。

**注 意**

---

この関数は、LS 中に一時バッファを 2 つ必要とします。このバッファは、呼び出し側関数 (`pLsInputTempBuffer`、`pLsOutputTempBuffer`) からパラメータとして指定することができます。

この関数は、`expectedUncompSize` が実際の伸張後のサイズと一致しない場合には、アサーションを生成します。

この関数に渡すのは、ヘッダのない生データへのポインタである必要があります。

この関数は、データエラーが発生した場合、(条件付き定義が変更されない限り) アサーションを発生する代わりに、呼出し元にエラー値を返します。



Filename: Edge\_Zlib-Library-Reference\_j.doc  
Directory: C:\Documents and Settings\MTESHIMA\My Documents\Working  
Files\7-19-10\Edge 1.2.0  
Template: C:\sony\yoshi\templates\libref\_V1.1\_j.dot  
Title: Edge\_Zlib\_Library-Reference\_j  
Subject:  
Author: SCE Document Group  
Keywords:  
Comments:  
Creation Date: 7/19/2010 3:34:00 PM  
Change Number: 3  
Last Saved On: 7/19/2010 3:34:00 PM  
Last Saved By: mteshima  
Total Editing Time: 1 Minute  
Last Printed On: 7/19/2010 3:35:00 PM  
As of Last Complete Printing  
Number of Pages: 48  
Number of Words: 12,980 (approx.)  
Number of Characters: 26,351 (approx.)