

# **PlayStation®Edge LZO**

## **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>
EdgeLzo1xDeflateQueueElement .....	6
EdgeLzo1xInflateQueueElement .....	8
EdgeLzo1xDeflateQHandle .....	10
EdgeLzo1xInflateQHandle .....	11
EdgeLzo1xDeflateTaskProcessing .....	12
EdgeLzo1xInflateTaskProcessing .....	13
EdgeLzo1xDeflateTaskProcessingStored .....	14
<b>PPU/SPU共有関数 .....</b>	<b>15</b>
edgeLzo1xAddDeflateQueueElement .....	16
edgeLzo1xTryAddDeflateQueueElement .....	18
edgeLzo1xAddInflateQueueElement .....	20
edgeLzo1xTryAddInflateQueueElement .....	22
edgeLzo1xAddInflateQueueElementPartialCopyOut .....	24
edgeLzo1xTryAddInflateQueueElementPartialCopyOut .....	26
<b>PPU関数 .....</b>	<b>29</b>
edgeLzo1xGetDeflateQueueSize .....	30
edgeLzo1xGetInflateQueueSize .....	31
edgeLzo1xCreateDeflateQueue .....	32
edgeLzo1xCreateInflateQueue .....	34
edgeLzo1xShutdownDeflateQueue .....	36
edgeLzo1xShutdownInflateQueue .....	37
edgeLzo1xGetDeflateTaskContextSaveSize .....	38
edgeLzo1xGetInflateTaskContextSaveSize .....	39
edgeLzo1xCreateDeflateTask .....	40
edgeLzo1xCreateInflateTask .....	42
<b>SPU関数 .....</b>	<b>44</b>
edgeLzo1xDeflateRawData .....	45
edgeLzo1xInflateRawData .....	46

# 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 LZ0 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、SPU 上で可逆データの伸長・圧縮を行うために使います。

## 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 クイックスタート

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 Zlib 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 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

# EdgeLzo1xDeflateQueueElement

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/lzo/edgelzo1x_deflate_queue_element.h>
typedef struct EdgeLzo1xDeflateQueueElement
{
    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_pad16;
} EdgeLzo1xDeflateQueueElement __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 警告：アドレスの下位 16 のビットは、返される圧縮データの内容に影響を与えます。これは、LZO1x 圧縮固有の性質です。
<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。
	最下位ビットは、 <a href="#">EdgeLzo1xDeflateTaskProcessing</a> 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、伸長が完了後にこの値に設定される
<i>m_pad16</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、このメモリを圧縮するための圧縮タスクを起動します。

## 注 意

圧縮後および圧縮前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。

圧縮前の入力データは、64K 以下である必要があります。

圧縮後の出力データは、 $m\_uncompressedSize + (m\_uncompressedSize/16) + 64 + 3$  以下になるはずで

警告：圧縮前の入力データのアドレスの下位 16 ビットは、lzo1x により、データのエンコード方法を決めるために使われるので、この下位 16 ビットを変更すると、エンコード後の圧縮データも（不正ではありませんが）異なるものになります。したがって、同一の圧縮されていないファイルを圧縮しても、その度に異なる圧縮データが生成される可能性があります。どの圧縮データも伸張後は同一データになります。

## **関 連 項 目**

---

[edgeLzo1xAddDeflateQueueElement](#), [edgeLzo1xCreateDeflateQueue](#)

# EdgeLzo1xInflateQueueElement

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/lzo/edgelzo1x_inflate_queue_element.h>
typedef struct EdgeLzo1xInflateQueueElement
{
    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;
} EdgeLzo1xInflateQueueElement __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。

伸張前のデータは、64K 以下である必要があります。

伸張後のデータは、64K 以下である必要があります。

`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

[edgeLzo1xAddInflateQueueElement](#), [edgeLzo1xAddInflateQueueElementPartialCopyOut](#), [edgeLzo1xCr  
eateInflateQueue](#)

---

# EdgeLzo1xDeflateQHandle

---

## 圧縮キューのハンドル

### 定 義 ( P P U の 場 合 )

---

```
#include <edge/lzo/edgelzo_ppu.h>
typedef void* EdgeLzo1xDeflateQHandle;
```

### 定 義 ( S P U の 場 合 )

---

```
#include <edge/lzo/edgelzo_spu.h>
typedef uint32_t EdgeLzo1xDeflateQHandle;
```

### 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

---

[EdgeLzo1xDeflateQueueElement](#), [edgeLzo1xCreateDeflateQueue](#)

---

# EdgeLzo1xInflateQHandle

---

## 伸長キューのハンドル

### 定 義 ( P P U の 場 合 )

---

```
#include <edge/lzo/edgelzo_ppu.h>
typedef void* EdgeLzo1xInflateQHandle;
```

### 定 義 ( S P U の 場 合 )

---

```
#include <edge/lzo/edgelzo_spu.h>
typedef uint32_t EdgeLzo1xInflateQHandle;
```

### 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

---

[EdgeLzo1xInflateQueueElement](#), [edgeLzo1xCreateInflateQueue](#)

# EdgeLzo1xDeflateTaskProcessing

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方を格納するかを指定する

## 定 義

```
#include <edge/lzo/edgelzo1x_deflate_queue_element.h>

typedef enum EdgeLzo1xDeflateTaskProcessing
{
    kEdgeLzo1xDeflateTask_DeflateStoreCompressed = 0,
    kEdgeLzo1xDeflateTask_DeflateStoreSmallest = 1,
} EdgeLzo1xDeflateTaskProcessing;
```

## メ ン バ

<i>kEdgeLzo1xDeflateTask_DeflateStoreCompressed</i>	常に圧縮されたデータを格納する
<i>kEdgeLzo1xDeflateTask_DeflateStoreSmallest</i>	圧縮前と圧縮後のデータのうち小さい方を格納する

## 解 説

この構造体は、圧縮によってデータのサイズが減らない場合に、それでも（圧縮後のデータの方が大きくても）圧縮後のデータを格納するか、それとも、小さい方のデータを格納するかを宣言するために使います。

## 関 連 項 目

[edgeLzo1xAddDeflateQueueElement](#)

---

# EdgeLzo1xInflateTaskProcessing

---

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/lzo/edgelzo1x_inflate_queue_element.h>

typedef enum EdgeLzo1xInflateTaskProcessing
{
    kEdgeLzo1xInflateTask_Memcpy = 0,
    kEdgeLzo1xInflateTask_Inflate = 1,
} EdgeLzo1xInflateTaskProcessing;
```

## メ ン バ

---

<i>kEdgeLzo1xInflateTask_Memcpy</i>	特定の場所から別の場所にメモリをコピーする
<i>kEdgeLzo1xInflateTask_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

---

[edgeLzo1xAddInflateQueueElement](#), [edgeLzo1xAddInflateQueueElementPartialCopyOut](#)

---

# EdgeLzo1xDeflateTaskProcessingStored

---

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/lzo/edgelzo1x_deflate_queue_element.h>

typedef enum EdgeLzo1xDeflateTaskProcessingStored
{
    kEdgeLzo1xDeflateTask_UncompressedWasStored = 0x00000000,
    kEdgeLzo1xDeflateTask_CompressedWasStored = 0x80000000,
} EdgeLzo1xDeflateTaskProcessingStored;
```

## メ ン バ

---

<i>kEdgeLzo1xDeflateTask_UncompressedWasStored</i>	格納されたのは圧縮前のデータ である
<i>kEdgeLzo1xDeflateTask_CompressedWasStored</i>	格納されたのは圧縮後のデータ である

## 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。

## 関 連 項 目

---

[edgeLzo1xAddDeflateQueueElement](#)

# PPU/SPU共有関数

# edgeLzo1xAddDeflateQueueElement

圧縮キューに新しい作業を追加する

## 定義 ( P P U の 場 合 )

```
#include <edge/lzo/edgelzo_ppu.h>
void edgeLzo1xAddDeflateQueueElement
(
    EdgeLzo1xDeflateQHandle handle,
    const void* eaInputUncompressedData,
    uint32_t uncompressedSize,
    void* eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t* eaOutputSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzo1xDeflateTaskProcessing processing
)
```

## 定義 ( S P U の 場 合 )

```
#include <edge/lzo/edgelzo_spu.h>
void edgeLzo1xAddDeflateQueueElement
(
    EdgeLzo1xDeflateQHandle handle,
    uint32_t eaInputUncompressedData,
    uint32_t uncompressedSize,
    uint32_t eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t eaOutputSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzo1xDeflateTaskProcessing 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。 警告：アドレスの下位 16 のビットは、返される圧縮データの内容に影響を与えます。これは、Lzo1x 圧縮固有の性質です。
<i>uncompressedSize</i>	圧縮前の入力データのサイズ
<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>	圧縮データバッファ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 <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。
<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。圧縮データにエラーがあった場合には、エラーが発生したという事実を警告するために、SPU は、このカウンタの最上位ビットを設定します。
<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
<i>processing</i>	圧縮完了後にイベントフラグはこの値に設定される。 常に圧縮データを格納するか、それとも、圧縮前と圧縮後のデータのうち小さい方を格納するかを選択します。
<i>dmaTag</i>	(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。

警告：圧縮前の入力データのアドレスの下位 16 ビットは、`lzolx` により、データのエンコード方法を決めるために使われるので、この下位 16 ビットを変更すると、エンコード後の圧縮データも（不正ではありませんが）異なるものになります。したがって、同一の圧縮されていないファイルを圧縮しても、その度に異なる圧縮データが生成される可能性があります。どの圧縮データも伸張後は同一データになります。

## 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。

`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

[EdgeLzolxDeflateQueueElement](#), [edgeLzolxCreateDeflateTask](#), [edgeLzolxTryAddDeflateQueueElement](#)

# edgeLzo1xTryAddDeflateQueueElement

圧縮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/lzo/edgelzo_ppu.h>
bool edgeLzo1xTryAddDeflateQueueElement
(
    EdgeLzo1xDeflateQHandle handle,
    const void* eaInputUncompressedData,
    uint32_t uncompressedSize,
    void* eaOutputCompressedData,
    uint32_t maxCompressedOutputSize,
    uint32_t* eaOutputCompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzo1xDeflateTaskProcessing processing
)
```

## 定 義 ( S P U の 場 合 )

```
#include <edge/lzo/edgelzo_spu.h>
bool edgeLzo1xTryAddDeflateQueueElement
(
    EdgeLzo1xDeflateQHandle 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,
    EdgeLzo1xDeflateTaskProcessing 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 警告：アドレスの下位 16 のビットは、返される圧縮データの内容に影響を与えます。これは、Lzo1x 圧縮固有の性質です。
<i>uncompressedSize</i>	圧縮前の入力データのサイズ
<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>eaOutputCompressedSize</i>	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、圧縮前のデータを格納することが選択されたかを示している
<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。圧縮データにエラーがあった場合には、PPU に警告を発するために、SPU は、このカウンタの最上位ビットを設定します。
<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
<i>processing</i>	圧縮完了後にイベントフラグはこの値に設定される 常に圧縮データを格納するか、それとも、圧縮前と圧縮後のデータのうち小さい方を格納するかを選択します。
<i>dmaTag</i>	(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

圧縮キューに要素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。追加された場合には、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。

警告：圧縮前の入力データのアドレスの下位 16 ビットは、`lzolx` により、データのエンコード方法を決めるために使われるので、この下位 16 ビットを変更すると、エンコード後の圧縮データも（不正ではありませんが）異なるものになります。したがって、同一の圧縮されていないファイルを圧縮しても、その度に異なる圧縮データが生成される可能性があります。どの圧縮データも伸張後は同一データになります。

## 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、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

[EdgeLzolxDeflateQueueElement](#), [edgeLzolxCreateDeflateTask](#), [edgeLzolxAddDeflateQueueElement](#)

# edgeLzo1xAddInflateQueueElement

伸長キューに新しい作業を追加する

## 定義 ( PPU の 場 合 )

```
#include <edge/lzo/edgelzo_ppu.h>
void edgeLzo1xAddInflateQueueElement
(
    EdgeLzo1xInflateQHandle handle,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzo1xInflateTaskProcessing processing
)
```

## 定義 ( SPU の 場 合 )

```
#include <edge/lzo/edgelzo_spu.h>
void edgeLzo1xAddInflateQueueElement
(
    EdgeLzo1xInflateQHandle handle,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzo1xInflateTaskProcessing 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>	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

<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。
<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 processing</i>	NULL であってもかまわない 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 <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>	(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

[EdgeLzolxInflateQueueElement](#), [edgeLzolxAddInflateQueueElementPartialCopyOut](#), [edgeLzolxCreateInflateTask](#)

# edgeLzo1xTryAddInflateQueueElement

伸長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/lzo/edgelzo_ppu.h>
bool edgeLzo1xTryAddInflateQueueElement
(
    EdgeLzo1xInflateQHandle handle,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzo1xInflateTaskProcessing processing
)
```

## 定義 ( SPU の 場 合 )

```
#include <edge/lzo/edgelzo_spu.h>
bool edgeLzo1xTryAddInflateQueueElement
(
    EdgeLzo1xInflateQHandle handle,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzo1xInflateTaskProcessing 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、エラーの発生を警告するために、そのエラーがカウンタの上位ビットを設定します。
<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> <i>processing</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 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>	(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

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* 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、*processing* に適切な値を渡せば、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

[EdgeLzo1xInflateQueueElement](#), [edgeLzo1xAddInflateQueueElementPartialCopyOut](#), [edgeLzo1xCreateInflateTask](#)

# edgeLzo1xAddInflateQueueElementPartialCopyOut

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/lzo/edgelzo_ppu.h>
void edgeLzo1xAddInflateQueueElementPartialCopyOut
(
    EdgeLzo1xInflateQHandle 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,
    EdgeLzo1xInflateTaskProcessing processing
)
```

## 定義 ( S P U の 場 合 )

```
#include <edge/lzo/edgelzo_spu.h>
void edgeLzo1xAddInflateQueueElementPartialCopyOut
(
    EdgeLzo1xInflateQHandle 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,
    EdgeLzo1xInflateTaskProcessing 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 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。
<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>	(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、`processing` に適切な値を渡せば、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、[edgeLzoxAddInflateQueueElement\(\)](#) 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

[EdgeLzoxInflateQueueElement](#), [edgeLzoxAddInflateQueueElement](#), [edgeLzoxCreateInflateTask](#)

# edgeLzo1xTryAddInflateQueueElementPartialCopyOut

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/lzo/edgelzo_ppu.h>
bool edgeLzo1xTryAddInflateQueueElementPartialCopyOut
(
    EdgeLzo1xInflateQHandle 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,
    EdgeLzo1xInflateTaskProcessing processing
)
```

## 定 義 ( S P U の 場 合 )

```
#include <edge/lzo/edgelzo_spu.h>
bool edgeLzo1xTryAddInflateQueueElementPartialCopyOut
(
    EdgeLzo1xInflateQHandle 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,
    EdgeLzo1xInflateTaskProcessing 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。

<i>compressedSize</i>	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 圧縮された入力データのサイズ
<i>eaOutputUncompPartialBuff</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>outputUncompSkipBeginSize</i>	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
<i>outputUncompPartialBuffSize</i>	DMA 転送される圧縮されていないデータのサイズ
<i>outputUncompSkipEndSize</i>	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
<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、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。
<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>	(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

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が、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、*processing* に適切な値を渡せば、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、[edgeLzo1xAddInflateQueueElement\(\)](#) 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**

---

[EdgeLzo1xInflateQueueElement](#), [edgeLzo1xAddInflateQueueElement](#), [edgeLzo1xCreateInflateTask](#)

# PPU関数

---

# edgeLzo1xGetDeflateQueueSize

---

圧縮キューに必要なバッファサイズを返す

## 定 義

---

```
#include <edge/lzo/edgelzo_ppu.h>
uint32_t edgeLzo1xGetDeflateQueueSize
(
    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

---

[EdgeLzo1xDeflateQueueElement](#), [edgeLzo1xCreateDeflateQueue](#), [edgeLzo1xCreateDeflateTask](#)

---

# edgeLzo1xGetInflateQueueSize

---

伸長キューに必要なバッファサイズを返す

## 定 義

---

```
#include <edge/lzo/edgeLzo_ppu.h>
uint32_t edgeLzo1xGetInflateQueueSize
(
    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

---

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

---

[EdgeLzo1xInflateQueueElement](#), [edgeLzo1xCreateInflateQueue](#), [edgeLzo1xCreateInflateTask](#)

# edgeLzo1xCreateDeflateQueue

## 圧縮キューを作成する

### 定 義

```
#include <edge/lzo/edgelzo_ppu.h>
EdgeLzo1xDeflateQHandle edgeLzo1xCreateDeflateQueue
(
    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
<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。 128 バイト境界にアラインメントされている必要がある メインメモリ上に提供されたバッファのサイズ。
<i>bufferSize</i>	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="#">edgeLzo1xGetDeflateQueueSize()</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 [EdgeLzo1xDeflateQHandle](#)

### 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

[EdgeLzo1xDeflateQueueElement](#), [edgeLzo1xAddDeflateQueueElement](#), [edgeLzo1xGetDeflateQueueSize](#), [edgeLzo1xShutdownDeflateQueue](#), [edgeLzo1xCreateDeflateTask](#)





# edgeLzo1xCreateInflateQueue

## 伸長キューを作成する

### 定 義

```
#include <edge/lzo/edgelzo_ppu.h>
EdgeLzo1xInflateQHandle edgeLzo1xCreateInflateQueue
(
    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>	この伸長キューが関連付けられる SPURS インスタンス。
<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。 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="#">edgeLzo1xGetInflateQueueSize()</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 [EdgeLzo1xInflateQHandle](#)

### 解 説

伸長キューを作成します。伸長キューは、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（伸長・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

[EdgeLzoInflateQueueElement](#), [edgeLzoAddInflateQueueElement](#), [edgeLzoAddInflateQueueElementPartialCopyOut](#), [edgeLzoGetInflateQueueSize](#), [edgeLzoShutdownInflateQueue](#), [edgeLzoCreateInflateTask](#)

---

# edgeLzo1xShutdownDeflateQueue

---

圧縮キューをシャットダウンする

## 定 義

---

```
#include <edge/lzo/edgelzo_ppu.h>
void edgeLzo1xShutdownDeflateQueue
(
    EdgeLzo1xDeflateQHandle 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

---

[edgeLzo1xCreateDeflateQueue](#)

---

# edgeLzo1xShutdownInflateQueue

---

伸長キューをシャットダウンする

## 定 義

---

```
#include <edge/lzo/edgelzo_ppu.h>
void edgeLzo1xShutdownInflateQueue
(
    EdgeLzo1xInflateQHandle 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

---

[edgeLzo1xCreateInflateQueue](#)

---

# edgeLzo1xGetDeflateTaskContextSaveSize

---

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/lzo/edgelzo_ppu.h>
uint32_t edgeLzo1xGetDeflateTaskContextSaveSize( 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

---

[edgeLzo1xCreateDeflateTask](#)

---

# edgeLzo1xGetInflateTaskContextSaveSize

---

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/lzo/edgelzo_ppu.h>
uint32_t edgeLzo1xGetInflateTaskContextSaveSize( 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

---

[edgeLzo1xCreateInflateTask](#)

# edgeLzo1xCreateDeflateTask

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/lzo/edgelzo_ppu.h>
CellSpursTaskId edgeLzo1xCreateDeflateTask
(
    CellSpursTaskset* pTaskSet,
    void* pTaskContext,
    EdgeLzo1xDeflateQHandle 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べき 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。 このバッファのために必要なサイズは、 <a href="#">edgeLzo1xGetDeflateTaskContextSaveSize()</a> を呼び出すことによって取得できる
<i>handle</i>	このタスクを取得す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、[edgeLzo1xGetDeflateTaskContextSaveSize\(\)](#) 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

[edgeLzo1xAddDeflateQueueElement](#), [edgeLzo1xGetDeflateQueueSize](#), [edgeLzo1xCreateDeflateQueue](#),  
[edgeLzo1xShutdownDeflateQueue](#), [edgeLzo1xGetDeflateTaskContextSaveSize](#)

# edgeLzo1xCreateInflateTask

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/lzo/edgelzo_ppu.h>
CellSpursTaskId edgeLzo1xCreateInflateTask
(
    CellSpursTaskset* pTaskSet,
    void* pTaskContext,
    EdgeLzo1xInflateQHandle 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。 このバッファのために必要なサイズは、 <a href="#">edgeLzo1xGetInflateTaskContextSaveSize</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、実行すべき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、[edgeLzo1xGetInflateTaskContextSaveSize\(\)](#)を呼び出すことによって取得できます。

## 関 連 項 目

[edgeLzo1xAddInflateQueueElement](#)、[edgeLzo1xAddInflateQueueElementPartialCopyOut](#)、[edgeLzo1xGetInflateQueueSize](#)、[edgeLzo1xCreateInflateQueue](#)、[edgeLzo1xShutdownInflateQueue](#)、[edgeLzo1xGetInflateTaskContextSaveSize](#)

# SPU関数

# edgeLzo1xDeflateRawData

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/lzo/edgelzo_spu.h>
extern int edgeLzo1xDeflateRawData
(
    const unsigned char* pUncompr,
    uint32_t uncompSize,
    unsigned char* pComprData,
    uint32_t maxCompressedDataSize,
    uint32_t* pOutputCompressedSize
);
```

## 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。 警告：アドレスの下位 16 のビットは、返される圧縮データの内容に影響を与えます。これは、Lzo1x 圧縮固有の性質です。
<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
<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

## 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 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、edgeLzo1xDeflateRawData 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。

警告：圧縮前の入力データのアドレスの下位 16 ビットは、lzo1x により、データのエンコード方法を決めるために使われるので、この下位 16 ビットを変更すると、エンコード後の圧縮データも（不正ではありませんが）異なるものになります。したがって、同一の圧縮されていないファイルを圧縮しても、その度に異なる圧縮データが生成される可能性があります。どの圧縮データも伸張後は同一データになります。

# edgeLzo1xInflateRawData

## 圧縮データのバッファを伸張する

### 定 義

```
#include <edge/lzo/edgelzo_spu.h>
extern int edgeLzo1xInflateRawData
(
    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、[edgeLzo1xInflateRawData](#) 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。

Filename: Edge\_LZO\_Library-Reference\_j.doc  
Directory: C:\Documents and Settings\MTESHIMA\My Documents\Working  
Files\7-19-10\Edge 1.2.0  
Template: C:\Documents and Settings\Administrator\My  
Documents\LAI\style templates\libref\_V1.1\_j.dot  
Title: Edge\_LZO\_Library-Reference\_j  
Subject:  
Author: SCE Document Group  
Keywords:  
Comments:  
Creation Date: 7/19/2010 3:29:00 PM  
Change Number: 2  
Last Saved On: 7/19/2010 3:29:00 PM  
Last Saved By: mteshima  
Total Editing Time: 2 Minutes  
Last Printed On: 7/19/2010 3:30:00 PM  
As of Last Complete Printing  
Number of Pages: 46  
Number of Words: 12,056 (approx.)  
Number of Characters: 24,234 (approx.)