

PlayStation®Edge LZMA 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
PPU/SPU共有ランタイムのデータ型	5
EdgeLzmaInflateQueueElement	6
EdgeLzmaInflateQHandle	8
EdgeLzmaInflateTaskProcessing	9
PPU/SPU共有関数	10
edgeLzmaAddInflateQueueElement	11
edgeLzmaTryAddInflateQueueElement	13
edgeLzmaAddInflateQueueElementPartialCopyOut	15
edgeLzmaTryAddInflateQueueElementPartialCopyOut	18
PPU関数	21
edgeLzmaGetInflateQueueSize	22
edgeLzmaCreateInflateQueue	23
edgeLzmaShutdownInflateQueue	24
edgeLzmaGetInflateTaskContextSaveSize	25
edgeLzmaCreateInflateTask	26
SPU関数	28
edgeLzmaInflateRawData	29

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 Edge LZMA 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 を利用した効率的なデータの解凍・移動を行うために使われます。

対象読者と前提条件

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 ディベロッパーのため書かれたものです。そのようなディベロッパーは、以下のような対象を熟知していることを前提にしています。

- C と 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 ライブラリ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 LZ0 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。
青字 + 下線	ハイパーリンクを表します（青色で表示されるのは、カラープリンタ、もしくはオンラインの場合だけです）。

PPU/SPU共有ランタイムのデータ型

EdgeLzmaInflateQueueElement

解凍・移動対象のデータセグメントを解凍タスクに知らせる解凍キュー内の要素

定 義

```
#include <edge/lzma/edgelzma_inflate_queue_element.h>
typedef struct EdgeLzmaInflateQueueElement
{
    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_properties0;
    uint16_t m_pad8;
} EdgeLzmaInflateQueueElement __attribute__((aligned(16)));
```

メ ン バ

<i>m_eaCompressed</i>	圧縮データの実効アドレス
<i>m_eaUncompressed</i>	解凍後データの実効アドレス
<i>m_compressedSize</i>	圧縮データのサイズ
<i>m_outputUncompPartialBuffSize</i>	DMA 転送される解凍後のデータのサイズ
<i>m_eaWorkToDoCounter</i>	解凍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。データ圧縮時に SPU でエラーが発生した場合には、エラーの発生を示すために、そのエラーがカウンタの最上位ビットに設定されます。 NULL であってもかまいません。
<i>m_eaEventFlag</i>	この実効アドレスの最下位ビットは、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)を示すフラグです イベントフラグの実効アドレス。 NULL であってもかまいません
<i>m_eventFlagBits</i>	イベントフラグは、解凍が完了後にこの値に設定されます
<i>m_outputUncompSkipBeginSize</i>	解凍後の出力のうち最初の <i>N</i> バイトを出力しない
<i>m_outputUncompSkipEndSize</i>	解凍後の出力のうち最後の <i>N</i> バイトを出力しない
<i>m_properties0</i>	プロパティデータの最初のバイト
<i>m_pad8</i>	無視されます

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、解凍タスクはメモリの解凍や移動を行います。

注 意

圧縮データ、および解凍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。

圧縮データは、64KB 以下である必要があります。

解凍後のデータは、64KB 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、圧縮データの解凍後の予想サイズが得られます。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

[edgeLzmaAddInflateQueueElement](#), [edgeLzmaAddInflateQueueElementPartialCopyOut](#), [edgeLzmaCreateInflateQueue](#)

EdgeLzmaInflateQHandle

解凍キューのハンドル

定義（ P P U の 場 合 ）

```
#include <edge/lzma/edgeLzma_ppu.h>
typedef void* EdgeLzmaInflateQHandle;
```

定義（ S P U の 場 合 ）

```
#include <edge/lzma/edgeLzma_spu.h>
typedef uint32_t EdgeLzmaInflateQHandle;
```

解 説

この型は、メインメモリ上の解凍キューを指すハンドルです。

関 連 項 目

[EdgeLzmaInflateQueueElement](#), [edgeLzmaCreateInflateQueue](#)

EdgeLzmaInflateTaskProcessing

セグメントをコピーするか解凍するかを指定します

定 義

```
#include <edge/lzma/edgeLzmaInflateQueueElement.h>

typedef enum EdgeLzmaInflateTaskProcessing
{
    kEdgeLzmaInflateTask_Memcpy = 0,
    kEdgeLzmaInflateTask_Inflate = 1,
} EdgeLzmaInflateTaskProcessing;
```

メ ン バ

<i>kEdgeLzmaInflateTask_Memcpy</i>	特定の場所から別の場所にメモリをコピーします
<i>kEdgeLzmaInflateTask_Inflate</i>	特定の場所から別の場所にメモリを解凍します

解 説

この型は、解凍タスクがセグメントをコピーするか解凍するかを宣言するために使われます。

関 連 項 目

[edgeLzmaAddInflateQueueElement](#), [edgeLzmaAddInflateQueueElementPartialCopyOut](#)

PPU/SPU共有関数

edgeLzmaAddInflateQueueElement

解凍キューに新しい作業を追加します

定義（ PPU の 場 合 ）

```
#include <edge/lzma/edgelzma_ppu.h>
void edgeLzmaAddInflateQueueElement
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzmaInflateTaskProcessing processing
)
```

定義（ SPU の 場 合 ）

```
#include <edge/lzma/edgelzma_spu.h>
void edgeLzmaAddInflateQueueElement
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzmaInflateTaskProcessing processing,
    uint32_t dmaTagId
)
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

引 数

<i>handle</i>	要素がプッシュされる解凍キューのハンドル
<i>pProperties</i>	プロパティデータへのポインタ これは、PPU 関数の場合には実効アドレス、SPU 関数の場合には LS （ローカルストア）のアドレスになります
<i>propertiesSize</i>	プロパティデータのサイズ

<i>eaInputCompressedData</i>	解凍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ります
<i>compressedSize</i>	圧縮状態の入力データのサイズ
<i>eaOutputUncompressed</i>	解凍後のデータ用の出力バッファの実効アドレス
<i>expectedUncompressedSize</i>	解凍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します
<i>eaWorkToDoCounter</i>	この要素の解凍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、エラーの発生を示すために、そのエラーがカウンタの最上位ビットに設定されます。
<i>eaEventFlag</i>	NULL であってもかまいません 設定するイベントフラグ
<i>eventFlagBits</i>	NULL であってもかまいません イベントフラグの設定に使われる値
<i>processing</i>	タスクがデータに対して行う処理の種類を選択します。 解凍を実行することも、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ます
<i>dmaTagId</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

この関数は解凍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、解凍タスクの 1 つに渡されます。

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最上位ビットに設定され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。

格納されるデータが圧縮されておらず、単に移動することだけが必要な場合にも、解凍タスクに *processing* 用の適切な値を渡せば、解凍タスクを利用して処理を行うことが可能です。

この関数は、プロパティデータへのポインタ、および生の圧縮データの実効アドレスの両方を受け取ります。

関 連 項 目

[EdgeLzmaInflateQueueElement](#), [edgeLzmaAddInflateQueueElementPartialCopyOut](#), [edgeLzmaCreateInflateTask](#)

edgeLzmaTryAddInflateQueueElement

解凍キューに新しい作業を追加することを試みます

定義（ PPU の 場 合 ）

```
#include <edge/lzma/edgelzma_ppu.h>
bool edgeLzmaTryAddInflateQueueElement
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    const void* eaInputCompressedData,
    uint32_t compressedSize,
    void* eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t* eaWorkToDoCounter,
    CellSpursEventFlag* eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzmaInflateTaskProcessing processing
)
```

定義（ SPU の 場 合 ）

```
#include <edge/lzma/edgelzma_spu.h>
bool edgeLzmaTryAddInflateQueueElement
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    uint32_t eaInputCompressedData,
    uint32_t compressedSize,
    uint32_t eaOutputUncompressed,
    uint32_t expectedUncompressedSize,
    uint32_t eaWorkToDoCounter,
    uint32_t eaEventFlag,
    uint16_t eventFlagBits,
    EdgeLzmaInflateTaskProcessing processing,
    uint32_t dmaTagId
)
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

引 数

<i>handle</i>	要素がプッシュされる解凍キューのハンドル
<i>pProperties</i>	プロパティデータへのポインタ。 これは、PPU 関数の場合には実効アドレス、SPU 関数の場合には LS （ローカルストア）のアドレスになります
<i>propertiesSize</i>	プロパティデータのサイズ
<i>eaInputCompressedData</i>	解凍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ります

<i>compressedSize</i>	圧縮状態の入力データのサイズ
<i>eaOutputUncompressed</i>	解凍後のデータ用の出力バッファの実効アドレス
<i>expectedUncompressedSize</i>	解凍後のデータの予想サイズ。
<i>eaWorkToDoCounter</i>	この値が誤っていると、SPU はアサーションを生成します この要素の解凍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。圧縮データの処理時に SPU でエラーが発生した場合には、エラーの発生を示すために、そのエラーがカウンタの最上位ビットに設定されます。
<i>eaEventFlag</i>	NULL であってもかまいません
<i>eventFlagBits</i>	設定するイベントフラグ。NULL であってもかまいません
<i>processing</i>	イベントフラグの設定に使われる値 タスクがデータに対して行う処理の種類を選択します。
	解凍を実行することも、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ます
<i>dmaTagId</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

この関数は、解凍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、解凍タスクの 1 つに渡されます。

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（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、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、それがカウンタの最上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。

格納されるデータが圧縮されておらず、単に移動することだけが必要な場合にも、解凍タスクに *processing* 用の適切な値を渡せば、解凍タスクを利用して処理を行うことが可能です。

この関数は、プロパティデータへのポインタ、および生の圧縮データの実効アドレスを受け取ります。

関 連 項 目

[EdgeLzmaInflateQueueElement](#), [edgeLzmaAddInflateQueueElementPartialCopyOut](#), [edgeLzmaCreateInflateQueue](#)

edgeLzmaAddInflateQueueElementPartialCopyOut

解凍キューに新しい作業を追加します

定義（ P P U の 場 合 ）

```
#include <edge/lzma/edgelzma_ppu.h>
void edgeLzmaAddInflateQueueElementPartialCopyOut
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    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,
    EdgeLzmaInflateTaskProcessing processing
)
```

定義（ S P U の 場 合 ）

```
#include <edge/lzma/edgelzma_spu.h>
void edgeLzmaAddInflateQueueElementPartialCopyOut
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    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,
    EdgeLzmaInflateTaskProcessing processing,
    uint32_t dmaTagId
)
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

引 数

<i>handle</i>	要素がプッシュされる解凍キューのハンドル
---------------	----------------------

<i>pProperties</i>	プロパティデータへのポインタ これは、PPU 関数の場合には実効アドレス、SPU 関数の場合には LS（ローカルストア）のアドレスになります
<i>propertiesSize</i>	プロパティデータのサイズ
<i>eaInputCompressedData</i>	解凍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ります
<i>compressedSize</i>	圧縮状態の入力データのサイズ
<i>eaOutputUncompPartialBuff</i>	解凍後のデータ用の出力バッファの実効アドレス
<i>outputUncompSkipBeginSize</i>	解凍後の出力のうち最初の N バイトを出力しない
<i>outputUncompPartialBuffSize</i>	DMA 転送される解凍後のデータのサイズ
<i>outputUncompSkipEndSize</i>	解凍後の出力のうち最後の N バイトを出力しない
<i>eaWorkToDoCounter</i>	この要素の解凍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、エラーの発生を示すために、そのエラーがカウンタの最上位ビットに設定されます。 NULL であってもかまいません
<i>eaEventFlag</i>	設定するイベントフラグ NULL であってもかまいません
<i>eventFlagBits</i>	イベントフラグの設定に使われる値
<i>processing</i>	タスクがデータに対して行う処理の種類を選択します。 解凍を実行することも、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ます
<i>dmaTagId</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

この関数は解凍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、解凍タスクの 1 つに渡されます。

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、それがカウンタの最上位ビットを設定する点に注意します。

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。

格納されるデータが圧縮されておらず、単に移動することだけが必要な場合にも、解凍タスクに *processing* 用の適切な値を渡せば、解凍タスクを利用してこの処理を行うことが可能です。

この関数は、プロパティデータへのポインタ、および生の圧縮データの実効アドレスの両方を受け取ります。

この関数は、[edgeLzmaAddInflateQueueElement\(\)](#)によく似ています。追加されたパラメータである *skipOutputBeginSize* および *skipOutputEndSize* は、セグメントの解凍は行えるが、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に存在しています。
outputUncompSkipBeginSize、*outputUncompPartialBuffSize*、*outputUncompSkipEndSize* を足すと、圧縮データの解凍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

[EdgeLzmaInflateQueueElement](#), [edgeLzmaAddInflateQueueElement](#), [edgeLzmaCreateInflateTask](#)

edgeLzmaTryAddInflateQueueElementPartialCopyOut

解凍キューに新しい作業を追加することを試みます

定義（ PPU の 場 合 ）

```
#include <edge/lzma/edgeLzma_ppu.h>
bool edgeLzmaTryAddInflateQueueElementPartialCopyOut
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    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,
    EdgeLzmaInflateTaskProcessing processing
)
```

定義（ SPU の 場 合 ）

```
#include <edge/lzma/edgeLzma_spu.h>
bool edgeLzmaTryAddInflateQueueElementPartialCopyOut
(
    EdgeLzmaInflateQHandle handle,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    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,
    EdgeLzmaInflateTaskProcessing processing,
    uint32_t dmaTagId
)
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

引 数

<i>handle</i>	要素がプッシュされる解凍キューのハンドル
---------------	----------------------

<i>pProperties</i>	プロパティデータへのポインタ これは、PPU 関数の場合には実効アドレス、SPU 関数の場合には LS（ローカルストア）のアドレスになります
<i>propertiesSize</i>	プロパティデータのサイズ
<i>eaInputCompressedData</i>	解凍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ります
<i>compressedSize</i>	圧縮状態の入力データのサイズ
<i>eaOutputUncompPartialBuff</i>	解凍後のデータ用の出力バッファの実効アドレス
<i>outputUncompSkipBeginSize</i>	解凍後の出力のうち最初の <i>N</i> バイトを出力しない
<i>outputUncompPartialBuffSize</i>	DMA 転送される解凍後のデータのサイズ
<i>outputUncompSkipEndSize</i>	解凍後の出力のうち最後の <i>N</i> バイトを出力しない
<i>eaWorkToDoCounter</i>	この要素の解凍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、エラーの発生を示すために、そのエラーがカウンタの最上位ビットに設定されます。
<i>eaEventFlag</i>	NULL であってもかまいません 設定するイベントフラグ
<i>eventFlagBits</i>	NULL であってもかまいません イベントフラグの設定に使われる値
<i>processing</i>	タスクがデータに対して行う処理の種類を選択します。 解凍を実行することも、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ます
<i>dmaTagId</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

この関数は、解凍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、解凍タスクの 1 つに渡されます。

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（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、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、それがカウンタの最上位ビットを設定する点に注意します。*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。

格納されるデータが圧縮されておらず、単に移動することだけが必要な場合にも、解凍タスクに *processing* 用の適切な値を渡せば、解凍タスクを利用してこの処理を行うことが可能です。

この関数は、プロパティデータへのポインタ、および生の圧縮データの実効アドレスの両方を受け取ります。

この関数は、[edgeLzmaAddInflateQueueElement](#)によく似ています。追加されたパラメータである *skipOutputBeginSize* および *skipOutputEndSize* は、セグメントの解凍は行えるが、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に存在しています。
outputUncompSkipBeginSize、*outputUncompPartialBuffSize*、*outputUncompSkipEndSize* を足すと、圧縮データの解凍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

[EdgeLzmaInflateQueueElement](#), [edgeLzmaAddInflateQueueElement](#), [edgeLzmaCreateInflateTask](#)

PPU関数

edgeLzmaGetInflateQueueSize

解凍キューが必要なバッファサイズを返します

定 義

```
#include <edge/lzma/edgeLzma_ppu.h>
uint32_t edgeLzmaGetInflateQueueSize
(
    uint32_t maxNumQueueEntries
)
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

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

この関数は、解凍キューが最大で指定された要素数を同時に保持できるように、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

[EdgeLzmaInflateQueueElement](#), [edgeLzmaCreateInflateQueue](#), [edgeLzmaCreateInflateTask](#)

edgeLzmaCreateInflateQueue

解凍キューを作成します

定 義

```
#include <edge/lzma/edgeLzma_ppu.h>
EdgeLzmaInflateQHandle edgeLzmaCreateInflateQueue
(
    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すことができます。
スレッドから呼び出すことができます（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す。

引 数

<i>pSpurs</i>	この解凍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>	この解凍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ります
<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、 edgeLzmaGetInflateQueueSize() を呼び出すことによって取得できます

返 り 値

作成された解凍キューの [EdgeLzmaInflateQHandle\(\)](#)。

解 説

この関数は解凍キューを作成します。解凍キューは、解凍タスクに処理させるためにキューに入れられた作業（解凍・移動するセグメント）のリストを保持します。

注 意

解凍キューは FIFO であり、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、キューのサイズは、実際に必要な最大要素数より小さくてもかまいません。

関 連 項 目

[EdgeLzmaInflateQueueElement](#), [edgeLzmaAddInflateQueueElement](#), [edgeLzmaAddInflateQueueElementPartialCopyOut](#), [edgeLzmaGetInflateQueueSize](#), [edgeLzmaShutdownInflateQueue](#), [edgeLzmaCreateInflateTask](#)

edgeLzmaShutdownInflateQueue

解凍キューをシャットダウンします

定 義

```
#include <edge/lzma/edgeLzma_ppu.h>
void edgeLzmaShutdownInflateQueue
(
    EdgeLzmaInflateQHandle handle
)
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

引 数

<i>handle</i>	シャットダウンする解凍キューのハンドル
---------------	---------------------

返 り 値

なし。

解 説

この関数は、解凍キューをシャットダウンします。

関 連 項 目

[edgeLzmaCreateInflateQueue](#)

edgeLzmaGetInflateTaskContextSaveSize

解凍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を返します

定 義

```
#include <edge/lzma/edgeLzma_ppu.h>
uint32_t edgeLzmaGetInflateTaskContextSaveSize( void )
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

引 数

（なし）

返 り 値

解凍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、解凍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

[edgeLzmaCreateInflateTask](#)

edgeLzmaCreateInflateTask

SPU 上で解凍を実行する SPURS タスクを作成します

定 義

```
#include <edge/lzma/edgelzma_ppu.h>
CellSpursTaskId edgeLzmaCreateInflateTask
(
    CellSpursTaskset* pTaskSet,
    void* pTaskContext,
    EdgeLzmaInflateQHandle handle
)
```

呼 出 条 件

割り込みハンドラから呼び出すことができます。
スレッドから呼び出すことができます（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す。

引 数

<i>pTaskSet</i>	この解凍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、 edgeLzmaGetInflateTaskContextSaveSize() を呼び出すことによって求めることができます
<i>handle</i>	このタスクが作業を取得する解凍キューのハンドル

返 り 値

作成されたタスクの CellSpursTaskId。

解 説

この関数は、解凍キューに作業が存在する場合に、解凍キューから作業を取り出す SPURS タスクを作成します。
実行する作業が存在しない場合には、タスクは、*pTaskContext* によって指定された場所にコンテキストを保存して、スリープします。

注 意

解凍タスクは、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 上で解凍を並列実行したい（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、同じ解凍キューに対して処理を行う、6 つの解凍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、[edgeLzmaGetInflateTaskContextSaveSize\(\)](#) 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

[edgeLzmaAddInflateQueueElement](#), [edgeLzmaAddInflateQueueElementPartialCopyOut](#), [edgeLzmaGetInflateQueueSize](#), [edgeLzmaCreateInflateQueue](#), [edgeLzmaShutdownInflateQueue](#), [edgeLzmaGetInflateTaskContextSaveSize](#)

SPU関数

edgeLzmaInflateRawData

圧縮データのバッファを解凍します

定 義

```
#include <edge/lzma/edgeLzma_spu.h>
extern int edgeLzmaInflateRawData
(
    unsigned char* pUncompr,
    uint32_t expectedUncompSize,
    const unsigned char* pProperties,
    uint32_t propertiesSize,
    const unsigned char* pComprData,
    uint32_t comprDataSize
);
```

引 数

<i>pUncompr</i>	ローカルストア内の、解凍後のデータが書き込まれる場所へのポインタ
<i>expectedUncompSize</i>	解凍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には、最低でもこの大きさが必要です
<i>pProperties</i>	ローカルストア内のプロパティデータへのポインタ
<i>propertiesSize</i>	プロパティデータのサイズ (バイト)
<i>pComprData</i>	ローカルストア内の、圧縮データが読み込まれる場所へのポインタ。 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ります
<i>comprDataSize</i>	読み込まれる圧縮データのバッファサイズ

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、ローカルストア中の圧縮データのバッファを解凍します。

注 意

入力バッファと出力バッファは、ともにローカルストアの中にある必要があります。[edgeLzmaInflateRawData\(\)](#) を呼び出す関数は、入力データを渡して、出力データを処理するものと仮定されます。この関数は、内部的にはDMAを行いません。

この関数は、*expectedUncompSize* が実際の解凍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。

この関数は、プロパティデータへのポインタ、および生の圧縮データのポインタを受け取ります。

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、呼出し元にエラーを返します。

Filename: Edge_LZMA_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\Jiangli Wu\My Documents\LAI
projects\templates\style templates\libref_V1.1_j.dot
Title: Edge_LZMA_Library-Reference_j
Subject:
Author: SCE Document Group
Keywords:
Comments:
Creation Date: 7/19/2010 3:14:00 PM
Change Number: 3
Last Saved On: 7/19/2010 3:15:00 PM
Last Saved By: mteshima
Total Editing Time: 3 Minutes
Last Printed On: 7/19/2010 3:17:00 PM
As of Last Complete Printing
Number of Pages: 29
Number of Words: 8,060 (approx.)
Number of Characters: 16,041 (approx.)