

データ圧縮ライブラリ

# RELC SO for Linux

custom01

ユーザーズマニュアル

01 版

富士通エレクトロニクス株式会社

## はじめに

### ■ 対象読者

本マニュアルは、C 言語の知識がある技術者の方を対象に記述しています。

### ■ 商標

RELC S0 for **Linux custom01** の著作権は富士通エレクトロニクス株式会社が保有しています。  
GNU Compiler Collection を GCC と表記しています。

その他、会社名、製品名の固有名詞は各社の商号、商標または登録商標です。

本マニュアルに記載されている会社名、システム名、製品名等には必ずしも商標表示 (TM・  
®) を付記していません。

1. 本資料に記載した製品および製品の仕様につきましては、製品改善のため予告なしに変更することがあります。したがって、ご使用を検討の際には、本資料に記載の情報が最新のものであることを弊社技術担当、あるいは弊社営業担当にご確認ください。
2. 本資料に記載された情報・回路図は、当社製品の応用例として使用されており、実際に使用する機器への搭載を目的としたものではありません。また、これらの情報・回路図の使用に起因する第三者の特許権、その他権利侵害について、当社はその責任を負いません。
3. 本資料に記載された製品は、一般事務用、パーソナル用、家庭用、通常の産業用等の一般的用途を想定して設計・製造されているものであり、原子力施設における核反応制御、航空機自動飛行制御、航空交通管制、大量輸送システムにおける運行制御、生命維持のための医療用機器、兵器システムにおけるミサイル発射制御など、極めて高度な安全性が要求され、仮に当該安全性が確保されない場合、直接生命・身体に対する重大な危険性を伴う用途（以下「ハイセイフティ用途」という）に使用されるよう設計・製造されたものではございません。お客様は、当該ハイセイフティ用途に要する安全性を確保する措置を施すことなく、本製品を使用しないでください。ハイセイフティ用途に使用される場合は、当社担当営業までご相談ください。
4. 本資料に記載した内容を、弊社に無断で転載または複製することはご遠慮ください。
5. 本資料に記載された製品が、「外国為替および外国貿易法」に基づき規制されている貨物または技術に該当する場合には、本製品を輸出するに際して、同法に基づく許可が必要となります。
6. 各社が提供する開発ツール（コンパイラおよび付属のライブラリ）の不具合および仕様に起因する問題については弊社の保証対象外とします。

本製品に含まれるライブラリは弊社の独自製品であり、GPL/LGPL 非適用ライブラリです。

本件について疑問点などありましたら、弊社担当営業までご相談ください

## 目次

1	概要	1
2	構成ファイル	1
3	開発環境	1
4	使用方法について	2
4.1	ライブラリおよびヘッダーファイルの配置	2
4.2	ヘッダーファイルのインクルード	2
4.3	コンパイラへ指定するオプション	2
4.4	ライブラリのリンク	3
5	API	4
5.1	制御情報領域	5
5.2	圧縮処理および復元処理の多重化	5
6	高レベル API	6
6.1	relc_HI_GetSizeOfInfo ()	8
6.2	relc_HI_FileEncode ()	9
6.3	relc_HI_FileDecode ()	10
6.4	relc_HI_Open ()	11
6.5	relc_HI_Write ()	12
6.6	relc_HI_Read ()	13
6.7	relc_HI_Close ()	14
6.8	relc_HI_GetAddInfoContents ()	15
6.9	relc_HI_GetAddInfo ()	16
6.10	高レベル API の使用方法	17
6.10.1	ファイルからファイルへの圧縮および復元	17
6.10.2	メモリからファイルへの圧縮	17
6.10.3	ファイルからメモリへの復元	18
7	中レベル API	19
7.1	relc_MI_GetSizeOfInfo	20
7.2	relc_MI_InitEncode	21
7.3	relc_MI_InitDecode	22
7.4	relc_MI_CallBack ()	23
7.5	relc_MI_Encode()	24
7.6	relc_MI_Decode ()	25
7.7	relc_MI_GetAddInfoContents ()	26
7.8	relc_MI_GetAddInfo ()	27
7.9	中レベル API の使用方法	28
7.9.1	圧縮処理	28
7.9.2	復元処理	29
8	低レベル API	30
8.1	relc_GetSizeOfInfo ()	31
8.2	relc_InitEncode ()	32
8.3	relc_InitDecode ()	33
8.4	relc_Encode ()	34

8.5	relc_EncodeFlush ()	35
8.6	relc_Decode ()	36
8.7	低レベル API の使用方法	37
8.7.1	圧縮処理	37
8.7.2	復元処理	38
9	定義	39
9.1	定数	39
9.2	型	39
10	エラーリファレンス	40
11	PC 版評価プログラムについて	41

## 1 概要

RELC SO for Linux custom01は、可逆圧縮方式(ロスレス圧縮方式)を使用してファイルやメモリに格納されたデータを圧縮するライブラリ製品です。

※RELCは Rapid Embedded Lossless Compression の略称です。

## 2 構成ファイル

本機能を利用するためには、次のライブラリとヘッダーファイルおよびシェルスクリプトを使用します。

librelc_mt.so.1.3.0	共有ライブラリ
relc_mt.h	低レベルAPI用ヘッダーファイル
relcm_mt.h	中レベルAPI用ヘッダーファイル
relch_mt.h	高レベルAPI用ヘッダーファイル
relctype.h	型定義ヘッダーファイル
makelink.sh	リンク作成シェルスクリプト

※製品パッケージの全体構成については、“README.TXT”を参照してください

## 3 開発環境

本製品は、GCC 4.1.2を使用して作成しています。

## 4 使用方法について

本製品の使用方法について説明します。

### 4.1 ライブラリおよびヘッダーファイルの配置

ライブラリを使用できる状態にするには、`package.tar` の内容をハードディスクへ取り出し、ライブラリへのシンボリックリンクを設定する必要があります。

ここでは配置の一例として、”`/tmp/package.tar`” の内容を”`/usr/local/lib/relc`”へ取り出し、”`/usr/lib`”へ配置する場合について説明します。配置操作を行うには、適切な権限が必要です。

1. `package.tar` の内容を`/usr/local/lib/relc`へ取り出します。

```
prompt%mkdir /usr/local/lib/relc
prompt%pushd /usr/local/lib/relc
prompt%tar xvf /tmp/package.tar
prompt%popd
```

2. `/usr/lib`へライブラリをコピーし、シンボリックリンクを作成します。

```
prompt%cp /usr/local/lib/relc/librelc_mt.so.* /usr/lib
prompt%pushd /usr/lib
prompt%sh /usr/local/lib/relc/makelink.sh
prompt%popd
```

3. `/sbin/ldconfig`を実行しライブラリキャッシュを更新します。

```
prompt%/sbin/ldconfig
```

### 4.2 ヘッダーファイルのインクルード

本製品のAPIを使用するソースファイルで、APIレベル（高、中、低）に応じたヘッダーファイルをインクルードしてください。

### 4.3 コンパイラーへ指定するオプション

本製品では、コンパイル時に次のオプションを指定しています。

```
-I 本製品を配置したディレクトリ(4.1の例では”/usr/local/lib/relc” )
-D__ANSI__
-D_LARGEFILE_SOURCE
-D_LARGEFILE64_SOURCE
```

#### 4.4 ライブラリのリンク

ライブラリのリンクには、リンク時に指定する方法とプログラム実行時にロードする方法があります。

- リンク時に指定する  
リンカオプションへ以下の指定を追加してください。

`-lrelc_mt`

- プログラム内からロードする  
プログラム内から `dlopen(3)` API を使用して、” `librelc_mt.so`” をロードし API アドレスを取得のうえで使用します。

詳しくは、開発環境のマニュアルを参照してください。

## 5 API

3つのレベル（高、中、低）のAPIを提供しています。レベルが高くなる程ライブラリ側が処理を負担し、呼び出し側のプログラムをシンプルに作成することができます。

各レベルで処理できる内容は、次のようになっています。

### ○ 高レベルAPI

- ・ メモリからファイルへの圧縮
- ・ ファイルからメモリへの復元
- ・ ファイルからファイルへの圧縮および復元
- ・ ヘッダー情報および付加情報の付加

### ○ 中レベルAPI

- ・ 任意サイズでのメモリからメモリへの圧縮および復元
- ・ ヘッダー情報および付加情報の付加

### ○ 低レベルAPI

- ・ 規定サイズでのメモリからメモリへの圧縮および復元

以降の説明では、区別のために圧縮前のデータをオリジナルデータ、圧縮後のデータを圧縮データと記述します。

圧縮を行う場合は、圧縮モードとして圧縮能力を数値（0もしくは1～128）で指定します。0（RELC\_MODE\_MAXIMUM）を指定すると、圧縮能力は最も高く（最高圧縮）になりますが、圧縮速度は低速です。128を指定すると圧縮能力はRELC\_MODE\_MAXIMUMより若干低下しますが、圧縮速度は若干高速になります。更に小さい値（RELC\_MODE\_HIGH, RELC\_MODE\_MID, RELC\_MODE\_LOW, または 128より小さい値）を指定すると、圧縮能力が低下する代わりに、圧縮速度は向上します。

次の定義を使用することができます。

RELC_MODE_MAXIMUM	圧縮モードを最高圧縮に設定します。(圧縮モード 0) 圧縮能力は最も高くなりますが、処理速度は最も遅くなります。
RELC_MODE_HIGH	圧縮モードを高圧縮に設定します。(圧縮モード 32)
RELC_MODE_MIDDLE	圧縮モードを中圧縮に設定します。(圧縮モード 16)
RELC_MODE_LOW	圧縮モードを低圧縮に設定します。(圧縮モード 4)
RELC_MODE_MINIMUM	圧縮モードを最低圧縮に設定します。(圧縮モード 1) 圧縮能力は最も低くなりますが、処理速度は最も速くなります。

※ この設定の有効性は、処理するオリジナルデータに依存します。オリジナルデータによっては、設定の効果が現れない場合があります。

中レベル以上の API では、圧縮データにヘッダー情報を付加することができます。ヘッダー情報を付加した場合は、さらに付加情報を付加することができますようになります。付加情報を付加することにより、復元の際にオリジナルデータのサイズを得るといったようなことができるようになります。ヘッダー情報は、必ず圧縮データの先頭に配置されている必要があります。

付加情報には、次のような種類があります。

<code>RELC_KIND_ZERO</code>	付加情報無し
<code>RELC_KIND_CODESIZE</code>	圧縮サイズ
<code>RELC_KIND_ORGSIZE</code>	オリジナルサイズ

圧縮サイズは、圧縮データの総サイズをバイト数で格納します。

オリジナルサイズは、オリジナルデータの総サイズをバイト数で格納します。

中レベルの API では、仕様上はリニアなメモリ空間への処理を想定しています。従って、有限サイズのバッファを介してそのバッファの大きさ以上の圧縮データやオリジナルデータを処理しようとするとはバッファリングが必要になります。また、付加情報を付ける場合に、`RELC_KIND_CODESIZE` といった事前に値が決定しない指定はできません。

高レベルの API では、ファイル操作を全てライブラリが行います。付加情報も全ての種類を指定できます。

## 5.1 制御情報領域

制御情報領域とは、一連のデータの処理の開始から完了まで処理状態を記録するための領域です。ユーザーは、使用するレベルの制御情報領域サイズ取得関数が返すバイト数分を確保してください。

## 5.2 圧縮処理および復元処理の多重化

圧縮および復元の処理を多重化する場合、並列処理したい一連のデータの数分制御情報領域が必要です。

## 6 高レベル API

このレベルでは、メモリからファイルへの圧縮、ファイルからメモリへの復元、ファイルからファイルへの圧縮および復元機能を提供します。

制御情報領域のサイズは、以下の API で取得します。

```
relc_HI_GetSizeOfInfo()
```

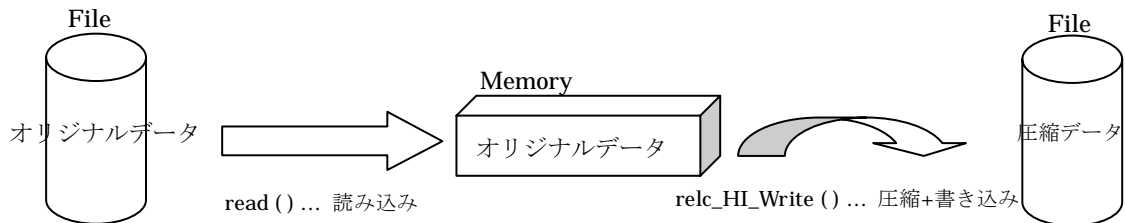
ファイルからファイルへの処理のために、次の 2 つの API を提供しています。

```
relc_HI_FileEncode()
```

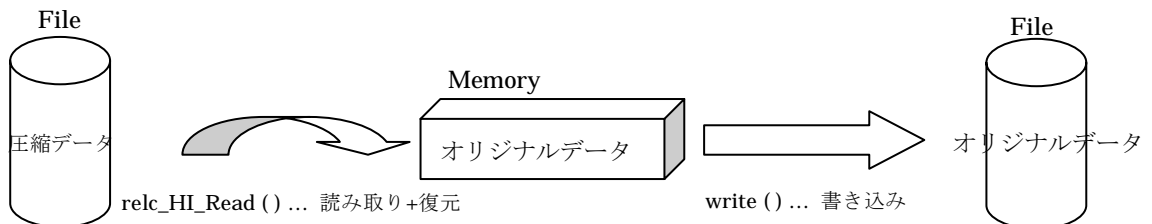
```
relc_HI_FileDecode()
```

ファイルからファイルへの圧縮または復元を行う場合は、`relc_HI_FileEncode()` と `relc_HI_FileDecode()` を使用します。

`relc_HI_FileEncode()` の操作イメージ



`relc_HI_FileDecode()` の操作イメージ

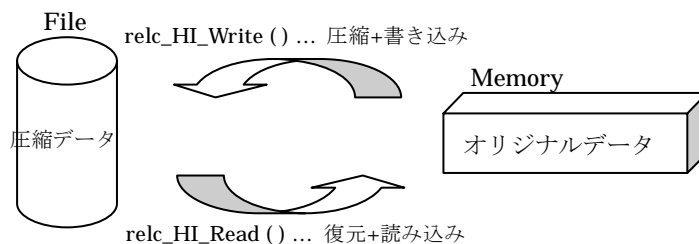


メモリからファイルへの圧縮処理とファイルからメモリへの復元処理のために、次の 6 つの API を提供しています。

```
relc_HI_Open()  
relc_HI_Write()  
relc_HI_Read()  
relc_HI_Close()  
relc_HI_GetAddInfoContents()  
relc_HI_GetAddInfo()
```

`relc_HI_Read()` は、ファイルから圧縮データを読み出し、復元したオリジナルデータをメモリに出力する処理を行います。また、`relc_HI_Write()` は、メモリからオリジナルデータを読み出し、圧縮した圧縮データをファイルに出力する処理を行います。

次に操作イメージを示します。



## 6.1 relc\_HI\_GetSizeOfInfo ( )

### [機能]

高レベル API を使用するために必要な制御情報領域のサイズを取得します

### [形式]

```
int relc_HI_GetSizeOfInfo (  
void  
)
```

### [引数]

無し

### [戻り値]

**int** == 制御情報領域のサイズ

### [説明]

戻り値として高レベル API 用の制御情報領域のサイズをバイト数で返します。

各高レベル API の第一引数へは、取得したサイズ分の領域へのポインタを指定します。

## 6.2 relc\_HI\_FileEncode ( )

### [機能]

ファイルを圧縮します。

### [形式]

```
int relc_HI_FileEncode (
void RELC_FAR *                pInfoArea ,
const char RELC_FAR *          pOriginalFileName ,
const char RELC_FAR *          pEncodedFileName ,
int                             nEncodeLevel ,
int                             bAddInfo
)
```

### [引数]

<b>pInfoArea</b>	高レベル API 用制御情報領域へのポインタ
<b>pOriginalFileName</b>	圧縮対象ファイルのファイル名が格納された領域へのポインタ
<b>pEncodedFileName</b>	圧縮データを格納するファイルのファイル名が格納された領域へのポインタ
<b>nEncodeLevel</b>	圧縮モード
<b>bAddInfo</b>	付加する情報の種類

### [戻り値]

<b>REL_CV_NORMAL</b>	==	正常終了
<b>REL_EV_ILLPARAM</b>	==	パラメータが異常
<b>REL_EV_FILEREAD</b>	==	ファイルの読み取りに失敗しました
<b>REL_EV_FILEWRITE</b>	==	ファイルの書き込みに失敗しました
<b>REL_EV_STOPUSER</b>	==	要求により処理を中断しました
<b>REL_EV_CLOSE</b>	==	ファイルのクローズでエラーがありました。

### [説明]

**pOriginalFileName** が示すファイルを圧縮し、圧縮データを **pEncodedFileName** に出力します。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

**nEncodeLevel** は圧縮モードを指定します。大きな数値ほど圧縮能力は高くなり、処理速度は低くなります。圧縮モードの指定方法については、「5 API」を参照してください。

**bAddInfo** は、付加するヘッダー情報や付加情報を指定します。

指定の際は、次の定数の論理和で指定します。

<b>REL_CKIND_NO_INFO</b>	ヘッダー情報無し
<b>REL_CKIND_ZERO</b>	付加情報無し
<b>REL_CKIND_ORGSIZE</b>	オリジナルサイズ
<b>REL_CKIND_CODESIZE</b>	圧縮サイズ

※REL\_CKIND\_NO\_INFO とそれ以外の定数を同時に指定しても、REL\_CKIND\_NO\_INFO 以外は無視されます。

### 6.3 relc\_HI\_FileDecode ( )

#### [機能]

ファイルを復元します。

#### [形式]

```
int relc_HI_FileDecode (
void RELC_FAR *          pInfoArea ,
const char RELC_FAR *    pEncodedFileName ,
const char RELC_FAR *    pOriginalFileName ,
int                      nIsAddInfo ,
int RELC_FAR *          *pbAddInfo
)
```

#### [引数]

<b>pInfoArea</b>	高レベル API 用制御情報領域へのポインタ
<b>pEncodedFileName</b>	圧縮データを格納するファイルのファイル名が格納された領域へのポインタ
<b>pOriginalFileName</b>	圧縮対象ファイルのファイル名が格納された領域へのポインタ
<b>nIsAddInfo</b>	ヘッダー情報の有無
<b>pbAddInfo</b>	付加情報の内容を格納する領域へのポインタ

#### [戻り値]

RELC_RV_NORMAL	==	正常終了
RELC_EV_ILLPARAM	==	パラメータが異常
RELC_EV_FILEREAD	==	ファイルの読み取りに失敗しました
RELC_EV_FILEWRITE	==	ファイルの書き込みに失敗しました
RELC_EV_STOPUSER	==	要求により処理を中断しました
RELC_EV_CLOSE	==	ファイルのクローズでエラーがありました。

#### [説明]

**pEncodedFileName** が示すファイルを復元し、復元データを **pOriginalFileName** に出力します。API の復帰時に **pnAddInfo** へ付加情報の内容を格納します。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

**nIsAddInfo** へは、付加情報の有無を指定します。**pEncodedFileName** が示すファイルに付加情報がある場合は 1 を無い場合は 0 を指定します。

**pbAddInfo** へ格納される値は、次の定数が論理和された値です

RELC_KIND_ZERO	付加情報無し
RELC_KIND_ORGSIZE	オリジナルサイズ
RELC_KIND_CODESIZE	圧縮サイズ

## 6.4 relc\_HI\_Open ( )

### [機能]

圧縮や復元のためにファイルをオープンします。

### [形式]

```
int relc_HI_Open (
void RELC_FAR *          pInfoArea ,
const char RELC_FAR *    pFilename ,
int                      nOpenMode ,
int                      nEncodeLevel ,
int                      bAddInfo
)
```

### [引数]

<b>pInfoArea</b>	高レベル API 用制御情報領域へのポインタ
<b>pFilename</b>	ファイル名の格納されている領域へのポインタ
<b>nOpenMode</b>	オープンするモード
<b>nEncodeLevel</b>	圧縮モード
<b>bAddInfo</b>	ヘッダー情報、付加情報の種類

### [戻り値]

<b>RELC_RV_NORMAL</b>	==	正常終了
<b>RELC_EV_CANTOPEN</b>	==	ファイルが開けません
<b>RELC_EV_ILLPARAM</b>	==	パラメータエラー

### [説明]

**pFilename** のファイルをオープンして圧縮または復元の準備をします。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

**nOpenMode** はオープンするモードを指定します。次の定数を使用してください。

<b>RELC_OPMD_ENCODE</b>	圧縮書き込みでファイルをオープン
<b>RELC_OPMD_DECODE</b>	復元読み出しでファイルをオープン

**nEncodeLevel** は圧縮モードを指定します。大きな数値ほど圧縮能力は高くなり、処理速度は低くなります。圧縮モードの指定方法については、「5 API」を参照してください。

**bAddInfo** は、付加するヘッダー情報や付加情報を指定します。

指定の際は、次の定数を使用します。

<b>RELC_KIND_NO_INFO</b>	ヘッダー情報無し
<b>RELC_KIND_ZERO</b>	付加情報無し
<b>RELC_KIND_ORGSIZE</b>	オリジナルサイズ
<b>RELC_KIND_CODESIZE</b>	圧縮サイズ

[圧縮の場合 (**nOpenMode** == **RELC\_OPMD\_ENCODE**) ]

定数を論理和した値を指定します。

※**RELC\_KIND\_NO\_INFO** とそれ以外の定数を論理和して同時に指定しても、**RELC\_KIND\_NO\_INFO** 以外は無視されます。

[復元の場合 (**nOpenMode** == **RELC\_OPMD\_DECODE**) ]

ヘッダー情報が付加されている圧縮ファイルをオープンする場合は 1、ヘッダー情報が付加されていない圧縮ファイルをオープンする場合は 0 を指定します。

## 6.5 relc\_HI\_Write ( )

### [機能]

指定されたメモリの内容を圧縮しファイルへ書き込みます。

### [形式]

```
int relc_HI_Write (
void RELC_FAR *          pInfoArea ,
char RELC_FAR *          pInBuf ,
long RELC_FAR *          pnInBuf
)
```

### [引数]

<b>pInfoArea</b>	高レベル API 用制御情報領域へのポインタ
<b>pInBuf</b>	オリジナルデータが格納されているバッファへのポインタ
<b>pnInBuf</b>	[ in ] バッファ内のオリジナルデータのサイズ ( 0x00000001 ~ 0x7FFFFFFF ) [ out ] 処理したオリジナルデータのサイズ ( 0x00000001 ~ 0x7FFFFFFF )

### [戻り値]

<b>RELC_RV_CONTINUE</b>	==	バッファ内のデータを使い切りました
<b>RELC_EV_FILEWRITE</b>	==	ファイルの書き込みに失敗しました
<b>RELC_EV_STOPUSER</b>	==	要求により処理を中断しました

### [説明]

**pInBuf** からオリジナルデータを読み取り、圧縮データをファイルへ書き込みます。読み取ったオリジナルデータのサイズは、**pnInBuf** に格納します。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

中レベルに定義されている **relc\_MI\_CallBack ( )** を使用してコールバック関数を設定することができます。コールバック関数を設定することにより呼び出し元から中断の指定を行えるようになります。詳しくは、「7.4 relc\_MI\_CallBack ( )」を参照してください。

## 6.6 relc\_HI\_Read ( )

### [機能]

指定されたメモリへファイルから読み込み復元します。

### [形式]

```
int relc_HI_Read (
void RELC_FAR *          pInfoArea ,
char RELC_FAR *          pOutBuf ,
long RELC_FAR *          pnOutBuf
)
```

### [引数]

pInfoArea	高レベル API 用制御情報領域へのポインタ
pOutBuf	オリジナルデータを出力するバッファへのポインタ
pnOutBuf	[ in ] バッファのサイズ ( 0x00000001 ~ 0x7FFFFFFF ) [ out ] 出力したオリジナルデータのサイズ ( 0x00000001 ~ 0x7FFFFFFF )

### [戻り値]

RELC_RV_CONTINUE	==	出力バッファを使い切りました
RELC_RV_ENDEDATA	==	全ての情報を復元しました
RELC_EV_FILEREAD	==	ファイルの読み取りに失敗しました
RELC_EV_ILLFORM	==	ファイル形式が正しくありません
RELC_EV_STOPUSER	==	要求により処理を中断しました

### [説明]

ファイルから圧縮データを読み取り **pOutBuf** へ書き込みます。書き込んだオリジナルデータのバイト数は、**pnOutBuf** に格納します。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

ファイルの内容を全て復元して **pOutBuf** へ出力できた場合は、RELC\_EV\_EDTEND を返します。

中レベルに定義されている **relc\_MI\_CallBack ( )** を使用してコールバック関数を設定しておくこと、コールバック関数内で呼び出し元から中断の指定をすることができます。詳しくは、「7.4 **relc\_MI\_CallBack ( )**」を参照してください。

## 6.7 relc\_HI\_Close ( )

### [機能]

ファイルをクローズします。

### [形式]

```
int relc_HI_Close (  
void RELC_FAR *          pInfoArea  
)
```

### [引数]

pInfoArea                    高レベル API 用制御情報領域へのポインタ

### [戻り値]

RELC\_RV\_NORMAL            == 正常終了  
RELC\_EV\_CLOSE             == ファイルのクローズでエラーがありました。

### [説明]

relc\_HI\_Open ( )でオープンしたファイルをクローズします。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

## 6.8 relc\_HI\_GetAddInfoContents ( )

### [機能]

圧縮データから付加情報の内容を得ます。

### [形式]

```
int relc_HI_GetAddInfoContents (  
void RELC_FAR *          pInfoArea ,  
int RELC_FAR *          pbAddInfo  
)
```

### [引数]

**pInfoArea**            高レベル API 用制御情報領域へのポインタ  
**pbAddInfo**            付加情報の種類を格納する領域へのポインタ

### [戻り値]

RELC\_RV\_NORMAL        ==    正常終了  
RELC\_EV\_ILLFORM       ==    フォーマットが正しくありません

### [説明]

relc\_HI\_Open ( )でオープンしたヘッダー情報付きの圧縮ファイルから付加情報を読み取り、**pbAddInfo** へ付加情報の種類を格納します。実際の付加情報の値を得るには、続けて relc\_HI\_GetAddInfo ( )を呼び出してください。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

**pbAddInfo** へ格納される値は、次の定数の論理和です。

RELC_KIND_ZERO	付加情報無し
RELC_KIND_ORGSIZE	オリジナルサイズ
RELC_KIND_CODESIZE	圧縮サイズ

## 6.9 relc\_HI\_GetAddInfo ( )

### [機能]

圧縮データから付加情報を得ます。

### [形式]

```
int relc_HI_GetAddInfo (  
void RELC_FAR *          pInfoArea ,  
int                      nAddInfo ,  
RELC_64VAR *            pnAddInfo  
)
```

### [引数]

<b>pInfoArea</b>	高レベル API 用制御情報領域へのポインタ
<b>nAddInfo</b>	取得する付加情報の種類
<b>pnAddInfo</b>	値を格納する領域へのポインタ

### [戻り値]

RELC_RV_NORMAL	==	正常終了
RELC_EV_ILLPARAM	==	パラメータが不正です

### [説明]

**nAddInfo** が示す圧縮ファイルの付加情報を読み取り、**pnAddInfo** へ格納します。

**pInfoArea** は高レベル用の制御情報領域へのポインタを指定します。

**nAddInfo** へは、次の値を指定します。

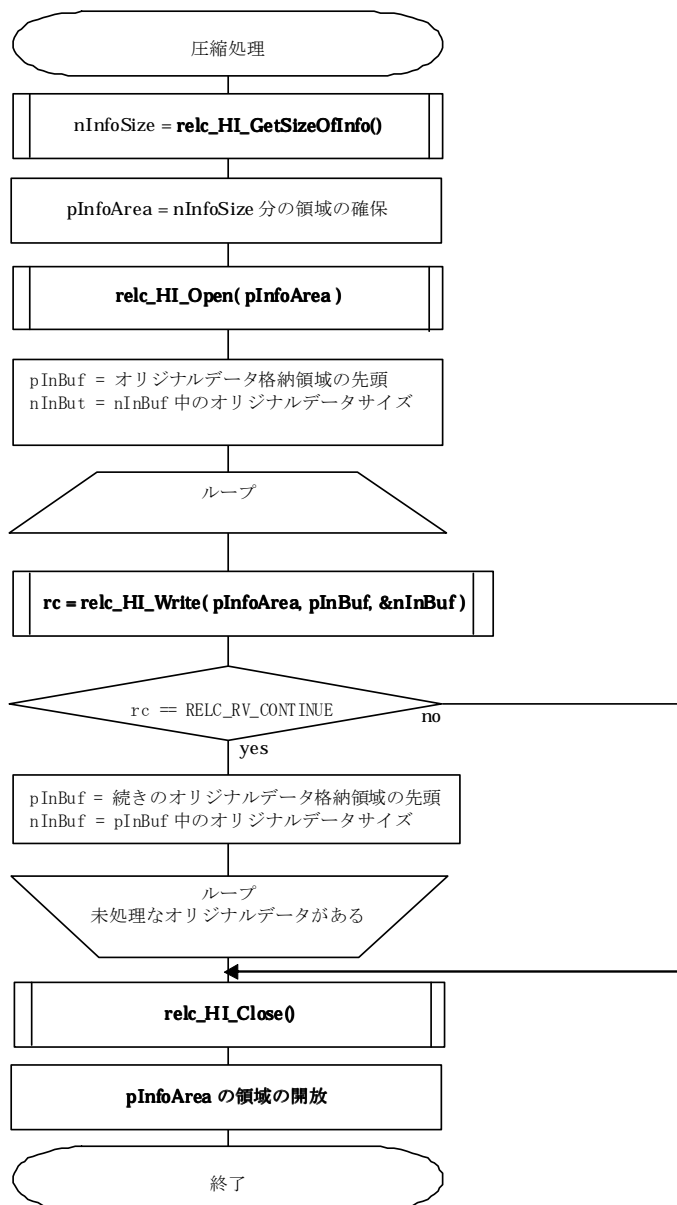
RELC_KIND_ORGSIZE	オリジナルデータサイズ
RELC_KIND_CODESIZE	圧縮データサイズ

## 6.10 高レベル API の使用方法

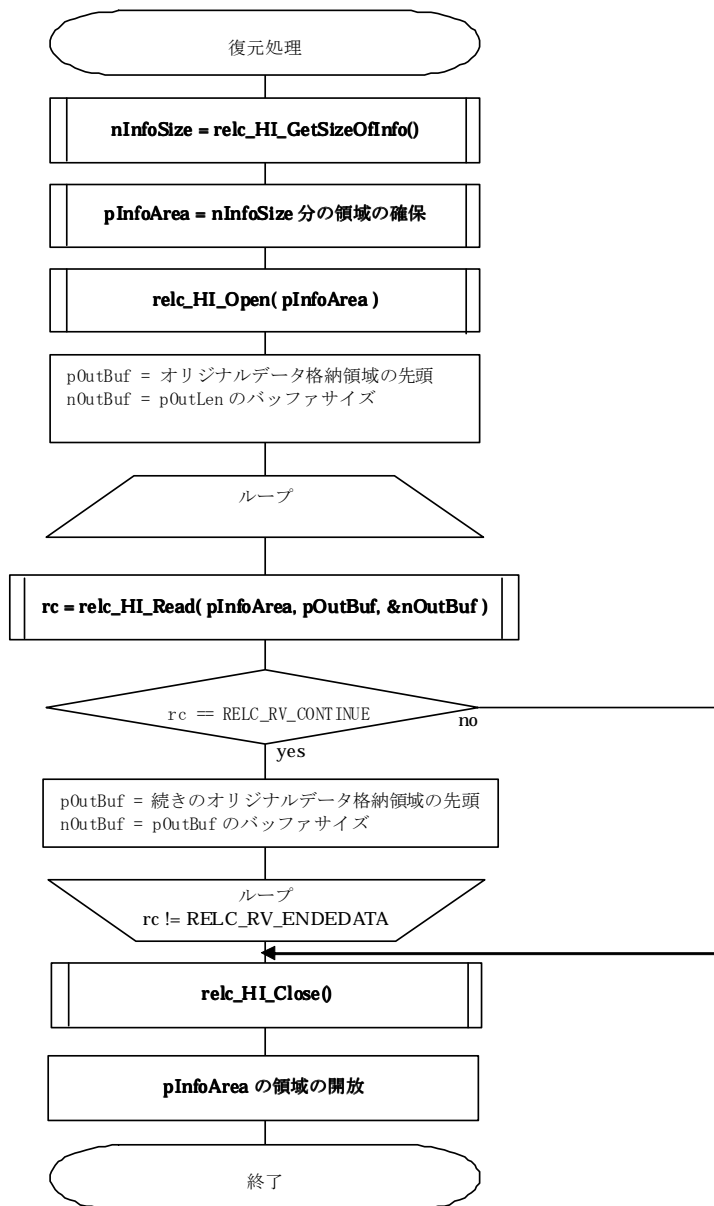
### 6.10.1 ファイルからファイルへの圧縮および復元

ファイル名指定で `rele_HI_FileEncode ( )` または、`rele_HI_FileDecode ( )` を呼び出してください。

### 6.10.2 メモリからファイルへの圧縮



### 6.10.3 ファイルからメモリへの復元



## 7 中レベル API

このレベルでは、任意のサイズ指定でのメモリからメモリへの圧縮および復元機能を提供します。また、復元処理の際に利用される付加情報を付ける機能も提供します。

次の 8 つの API を提供しています。

```
relc_MI_GetSizeOfInfo()
relc_MI_InitEncode()
relc_MI_InitDecode()
relc_MI_CallBack()
relc_MI_Encode()
relc_MI_Decode()
relc_MI_GetAddInfoContents()
relc_MI_GetAddInfo()
```

制御情報領域のサイズは、以下の API で取得します。

```
relc_MI_GetSizeOfInfo()
```

圧縮処理における関数の呼び出しは、次の順序で行います。

```
relc_MI_InitEncode() → relc_MI_Encode()
```

復元処理における関数の呼び出しは、次の順序で行います。

```
relc_MI_InitDecode() → relc_MI_Decode()
```

このレベルの API の戻り値は、制御フィールドとエラーフィールドに分かれています。エラーコードを得るには、戻り値と **RELC\_MEV\_MASK** を論理積してください。

制御フィールドはビットフィールドになっています。そのため、戻り値と次の値を論理積した結果が真の場合は、その状態に応じた処理を行ってください。

<b>RELC_MCV_NODATA</b>	入力データを全て処理しました。次の入力データを与えてください
<b>RELC_MCV_NOBUFF</b>	出力バッファを全て使用しました。新しい出力バッファを用意してください。
<b>RELC_MCV_EDTEND</b>	全ての処理が終了しました。終了処理を行ってください。

## 7.1 relc\_MI\_GetSizeOfInfo

### [機能]

中レベル API を使用するために必要な制御情報領域のサイズを取得します。

### [形式]

```
int relc_MI_GetSizeOfInfo(  
void  
)
```

### [引数]

無し

### [戻り値]

**int** == 制御情報領域のサイズ

### [説明]

戻り値として中レベル API 用の制御情報領域のサイズをバイト数で返します。

各中レベル API の第一引数へは、取得したサイズ分の領域へのポインタを指定します。

## 7.2 relc\_MI\_InitEncode

### [機能]

圧縮処理のための準備を行います。

### [形式]

```
int relc_MI_InitEncode(
void RELC_FAR *      pInfoArea ,
int                  nEncodeLevel ,
int                  nAddInfo ,
RELC_64VAR           nOriginalSize
)
```

### [引数]

<b>pInfoArea</b>	中レベル API 用制御情報領域へのポインタ
<b>nEncodeLevel</b>	圧縮モード
<b>nAddInfo</b>	付加情報の種類
<b>nOriginalSize</b>	オリジナルサイズ

### [戻り値]

エラーフィールド (戻り値 & **RELC\_MEV\_MASK**)

<b>RELC_RV_NORMAL</b>	==	正常終了
<b>RELC_EV_ILLPARAM</b>	==	圧縮モード指定エラー

制御フィールド (戻り値 & **RELC\_MCV\_XXXX**)  
無し

### [説明]

一連のオリジナルデータを **relc\_Encode( )** で処理する前に呼び出してください。引数として **nEncodeLevel** へ圧縮モード、**nAddInfo** へ付加情報の種類、**nOriginalSize** へオリジナルデータの総バイト数を指定します。

**pInfoArea** は中レベル用の制御情報領域へのポインタを指定します。

**nEncodeLevel** は圧縮モードを指定します。大きな数値ほど圧縮能力は高くなり、処理速度は低くなります。圧縮モードの指定方法については、「5 API」を参照してください。

**nAddInfo** は、次の定数を使用してください。

<b>RELC_KIND_NO_INFO</b>	ヘッダー情報無し
<b>RELC_KIND_ZERO</b>	付加情報無し
<b>RELC_KIND_ORGSIZE</b>	オリジナルサイズ

**nOriginalSize** へは圧縮しようとするデータの総バイト数を指定してください。

### 7.3 relc\_MI\_InitDecode

#### [機能]

復元処理のための準備を行います。

#### [形式]

```
int relc_MI_InitDecode(  
void RELC_FAR *      pInfoArea ,  
int                  nAddInfo  
)
```

#### [引数]

**pInfoArea**            中レベル API 用制御情報領域へのポインタ  
**nAddInfo**            ヘッダー情報の有無

#### [戻り値]

エラーフィールド        ( 戻り値 & RELC\_MEV\_MASK )  
RELCV\_NORMAL            ==    正常終了

制御フィールド        ( 戻り値 & RELC\_MCV\_XXXX )  
無し

#### [説明]

一連の圧縮データをrelc\_Decode( )で処理する前に呼び出してください。**nIsAddInfo**はヘッダー情報の有無を指定します。復元する圧縮データにヘッダー情報がある場合は1を無い場合は0を指定します。

**pInfoArea** は中レベル用の制御情報領域へのポインタを指定します。

## 7.4 relc\_MI\_CallBack ( )

### [機能]

コールバック関数を設定します。

### [形式]

```
into relc_MI_CallBack(  
void RELC_FAR *      pInfoArea ,  
void RELC_FAR *      pCallback  
)
```

### [引数]

**pInfoArea**            中レベル API 用制御情報領域へのポインタ  
**pCallback**           コールバック関数へのポインタ

### [戻り値]

エラーフィールド (戻り値 & RELC\_MEV\_MASK)  
RELC\_RV\_NORMAL        ==    正常終了

制御フィールド (戻り値 & RELC\_MCV\_XXXX)  
無し

### [説明]

中レベルの API は、低レベルの API を連続して呼び出します。そのため大きなデータを処理させると、長い時間処理が終わらない場合があります。

**pInfoArea** は中レベル用の制御情報領域へのポインタを指定します。

この API を使用してコールバック関数を設定しておくこと、低レベル API を 1 回呼び出すごとにセットされた関数を呼び出します。

コールバック関数は、次のように定義されている必要があります。

```
void ( RELC_FAR * pCallback)( int RELC_FAR * );
```

ライブラリは、ライブラリ内部の処理中断用の変数を引数としてコールバック関数を呼び出します。この引数に 0 以外の数値を代入することで処理を中断することができます。

## 7.5 relc\_MI\_Encode( )

### [機能]

オリジナルデータを圧縮します。

### [形式]

```
int relc_MI_Encode (
void RELC_FAR *          pInfoArea ,
unsigned char RELC_FAR * pInBuf ,
unsigned char RELC_FAR * pOutBuf ,
long RELC_FAR *         pnInBuf ,
long RELC_FAR *         pnOutBuf
)
```

### [引数]

<b>pInfoArea</b>	中レベル API 用制御情報領域へのポインタ
<b>pInBuf</b>	オリジナルデータの入力バッファアドレス
<b>pOutBuf</b>	圧縮データの出力バッファアドレス
<b>pnInBuf</b>	[in ] 入力バッファ中のオリジナルデータのバイト数 ( 0x00000000~0x7FFFFFFF )
	[out] 入力バッファ中で使用したオリジナルデータのバイト数
<b>pnOutBuf</b>	[in ] 出力バッファのサイズ ( 0x00008002~0x7FFFFFFF )
	[out] 出力バッファへ出力された圧縮データのバイト数

### [戻り値]

エラーフィールド ( 戻り値 & RELC\_MEV\_MASK )

RELc\_RV\_NORMAL == 正常終了

RELc\_EV\_STOPUSER == 処理を中断

制御フィールド ( 戻り値 & RELc\_MCV\_XXXX )

RELc\_MCV\_NODATA 入力データを使い切りました

RELc\_MCV\_NOBUFF 出力バッファを使い切りました

RELc\_MCV\_EDTEND 圧縮処理が終了

### [説明]

**pInBuf** から **pnInBuf** バイトのオリジナルデータを読み取り、圧縮データを **pOutBuf** へ書き込みます。関数から復帰すると、**pnInBuf** へ使用されたオリジナルデータのバイト数、**pnOutBuf** へ出力された圧縮データのバイト数が格納されています。

**pInfoArea** は中レベル用の制御情報領域へのポインタを指定します。

オリジナルデータを格納する入力バッファは、RELc\_MAX\_ORIGINAL バイト以上のサイズを用意してください。

圧縮データを格納する出力バッファは、RELc\_MAX\_CODE バイト以上のサイズを用意してください。

**pInBuf** の領域への操作は読み取りのみで書き込みは行いません。

**pOutBuf** の領域への操作は書き込みのみで読み取りは行いません。

## 7.6 relc\_MI\_Decode ()

### [機能]

圧縮データを復元します。

### [形式]

```
int relc_MI_Decode(
void RELC_FAR *          pInfoArea ,
unsigned char RELC_FAR * pInBuf ,
unsigned char RELC_FAR * pOutBuf ,
long RELC_FAR *         pnInLen ,
long RELC_FAR *         pnOutLen
)
```

### [引数]

pInfoArea	中レベル API 用制御情報領域へのポインタ
pInBuf	圧縮データの入力バッファアドレス
pOutBuf	オリジナルデータの出力バッファアドレス
pnInLen	[in ] 入力バッファ中の圧縮データのバイト数 ( 0x00000000 ~ 0x7FFFFFFF )
	[out] 入力バッファで使った圧縮データのバイト数
pnOutLen	[in ] 出力バッファサイズ ( 0x00008000 ~ 0x7FFFFFFF )
	[out] 出力バッファへ出力したバイト数

### [戻り値]

エラーフィールド ( 戻り値 & RELC\_MEV\_MASK )

RELC_RV_NORMAL	==	正常終了
RELC_EV_STOPUSER	==	処理を中断

### 制御フィールド ( 戻り値 & RELC\_MCV\_XXXX )

RELC_MCV_NODATA	入力データを使い切りました
RELC_MCV_NOBUFF	出力バッファを使い切りました
RELC_MCV_EDTEND	圧縮データが終了

### [説明]

**pInBuf** から圧縮データを読み取り、復元したオリジナルデータを **pOutBuf** へ書き込みます。関数から復帰すると、**pnInLen** へ読み取った圧縮データのバイト数、**pnOutLen** へ書き込んだオリジナルデータのバイト数が格納されています。

**pInfoArea** は中レベル用の制御情報領域へのポインタを指定します。

圧縮データを格納する入力バッファは、RELC\_MAX\_CODE バイト以上のサイズを用意してください。

オリジナルデータを格納する出力バッファは、RELC\_MAX\_ORIGINAL バイト以上のサイズを用意してください。

**pInBuf** の領域への操作は読み取りのみで書き込みは行いません。

**pOutBuf** の領域への操作は読み書きを行います。

## 7.7 relc\_MI\_GetAddInfoContents ( )

### [機能]

圧縮データから付加情報の内容を得ます。

### [形式]

```
int relc_MI_GetAddInfoContents (  
void RELC_FAR *          pInfoArea ,  
unsigned char RELC_FAR *  pInBuff ,  
long RELC_FAR *          pnInLen ,  
int RELC_FAR *           pbAddInfo ,  
)
```

### [引数]

<b>pInfoArea</b>	中レベル API 用制御情報領域へのポインタ
<b>pInBuff</b>	圧縮データの入力バッファアドレス
<b>pnInLen</b>	入力バッファ中の圧縮データのバイト数
<b>pbAddInfo</b>	付加情報の種類

### [戻り値]

エラーフィールド (戻り値 & RELC\_MEV\_MASK)  
RELC\_RV\_NORMAL == 正常終了  
RELC\_EV\_ILLFORM == フォーマットが不正です

制御フィールド (戻り値 & RELC\_MCV\_XXXX)  
無し

### [説明]

**pnInBuff** の圧縮データから付加情報を読み取り、**pbAddInfo** へ付加情報の種類を格納します。実際の付加情報の値を得るには、続けて `relc_MI_GetAddInfo ( )` を呼び出してください。

**pInfoArea** は中レベル用の制御情報領域へのポインタを指定します。

**pbAddInfo** へ格納される値は、次の定数の論理和です。

RELC_KIND_NO_INFO	情報無し
RELC_KIND_ORGSIZE	オリジナルサイズ
RELC_KIND_CODESIZE	圧縮サイズ

## 7.8 relc\_MI\_GetAddInfo ( )

### [機能]

圧縮データの付加情報を読み取ります。

### [形式]

```
int relc_MI_GetAddInfo (
void RELC_FAR *          pInfoArea ,
int                      nAddInfo ,
RELC_64VAR *            pnAddInfo
)
```

### [引数]

<b>pInfoArea</b>	中レベル API 用制御情報領域へのポインタ
<b>nAddInfo</b>	取得する付加情報
<b>pnAddInfo</b>	値を格納する領域へのポインタ

### [戻り値]

エラーフィールド	( 戻り値 & RELC_MEV_MASK )
RELC_RV_NORMAL	== 正常終了
RELC_EV_ILLFORM	== フォーマットエラー

制御フィールド	( 戻り値 & RELC_MCV_XXXX )
無し	

### [説明]

**nAddInfo** で指定された付加情報を圧縮データの先頭から読み取り、**pnAddInfo** へ格納します。

**pInfoArea** は中レベル用の制御情報領域へのポインタを指定します。

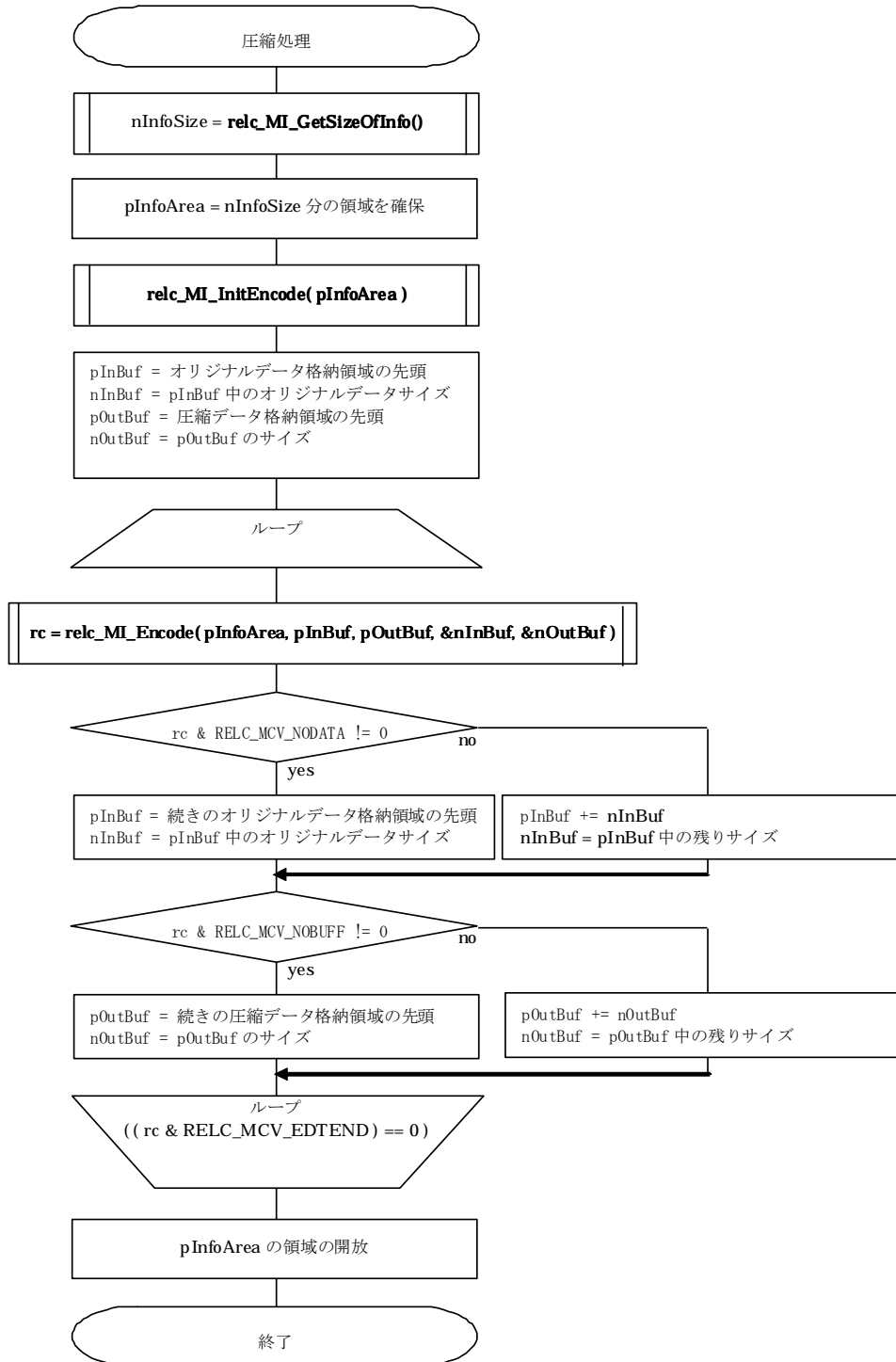
**nAddInfo** へは、次の値を指定します。

RELC_KIND_ORGSIZE	オリジナルサイズ
RELC_KIND_CODESIZE	圧縮サイズ

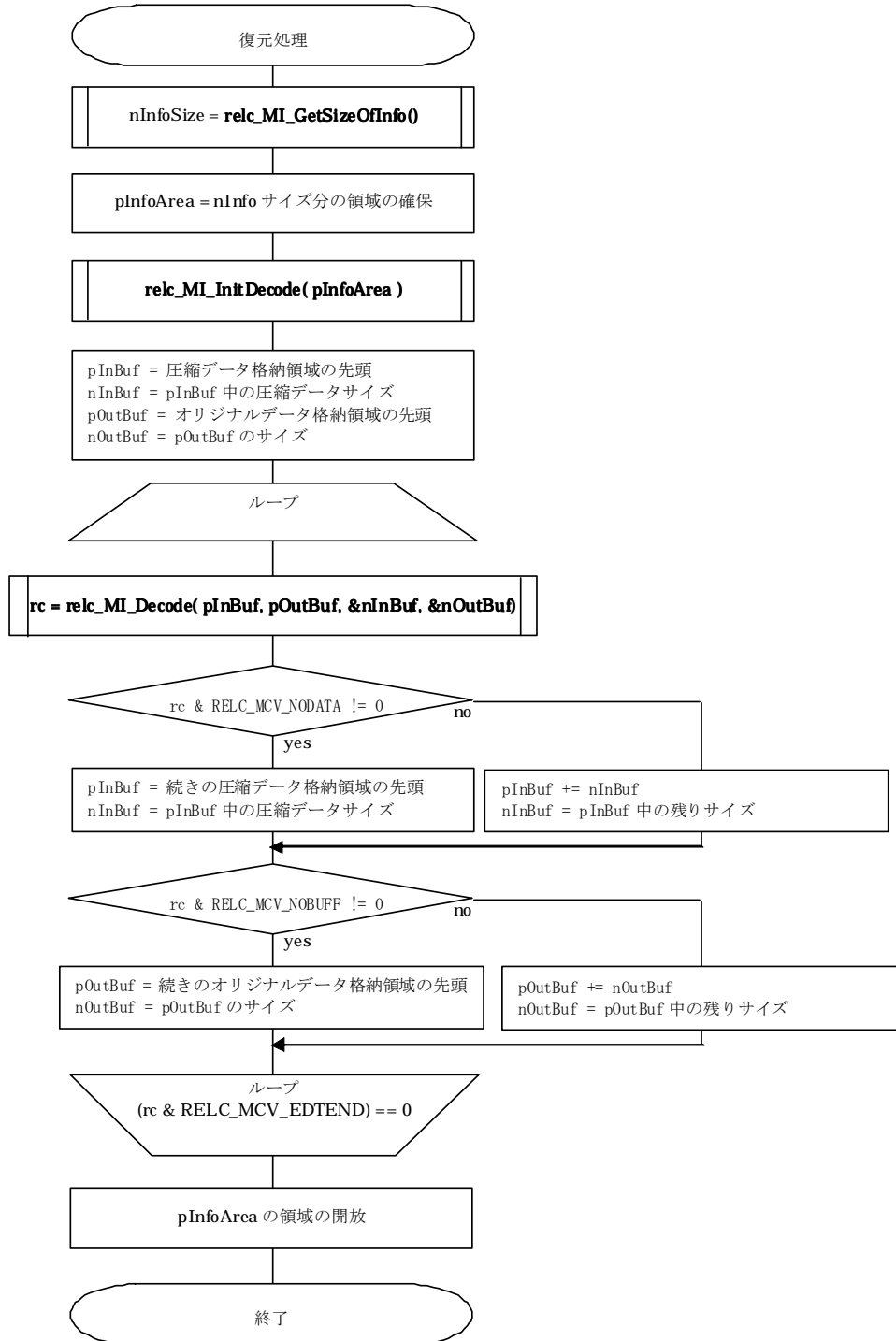
## 7.9 中レベル API の使用方法

このレベルでは、メモリからメモリへの圧縮および復元を提供しています。

### 7.9.1 圧縮処理



## 7.9.2 復元処理



## 8 低レベル API

このレベルでは、最も基本的なメモリからメモリへの圧縮および復元機能を提供しています。1回の呼出で処理できるサイズは規定されているため、それ以上のサイズを処理する場合は、バッファリングが必要となります。

次の 6 つの API を提供しています。

```
relc_GetSizeOfInfo()
relc_InitEncode()
relc_InitDecode()
relc_Encode()
relc_EncodeFlush()
relc_Decode()
```

制御情報領域のサイズは、以下の API で取得します。

```
relc_GetSizeOfInfo()
```

圧縮処理とは、入力アドレスが示すメモリに格納されたオリジナルデータを読み出し、出力アドレスが示すメモリへ圧縮データを書き出す処理です。圧縮処理における関数の呼び出しは、次の順序で行います。

```
relc_InitEncode() → relc_Encode() → relc_EncodeFlush()
```

復元処理とは、入力アドレスが示すメモリに格納された圧縮データを読み出し、出力アドレスが示すメモリへオリジナルデータを書き出す処理です。復元処理における関数の呼び出しは、次の順序で行います。

```
relc_InitDecode() → relc_Decode()
```

## 8.1 relc\_GetSizeOfInfo ( )

### [機能]

低レベル API を使用するために必要な制御情報領域のサイズを取得します。

### [形式]

```
int relc_GetSizeOfInfo(  
void  
)
```

### [引数]

無し

### [戻り値]

```
int          ==      正常終了
```

### [説明]

戻り値として低レベル API 用の制御情報領域のサイズをバイト数で返します。

各低レベル API へは、取得したサイズ分の領域へのポインタを指定します。

## 8.2 relc\_InitEncode ( )

### [機能]

圧縮処理のための準備を行います。

### [形式]

```
int relc_InitEncode(
void RELC_FAR *      pInfoArea ,
int                  nEncodeLevel
)
```

### [引数]

**pInfoArea**            低レベル API 用制御情報領域へのポインタ  
**nEncodeLevel**        圧縮モード

### [戻り値]

**RELC\_RV\_NORMAL**    ==    正常終了  
**RELC\_EV\_ILLPARAM** ==    圧縮モード指定エラー

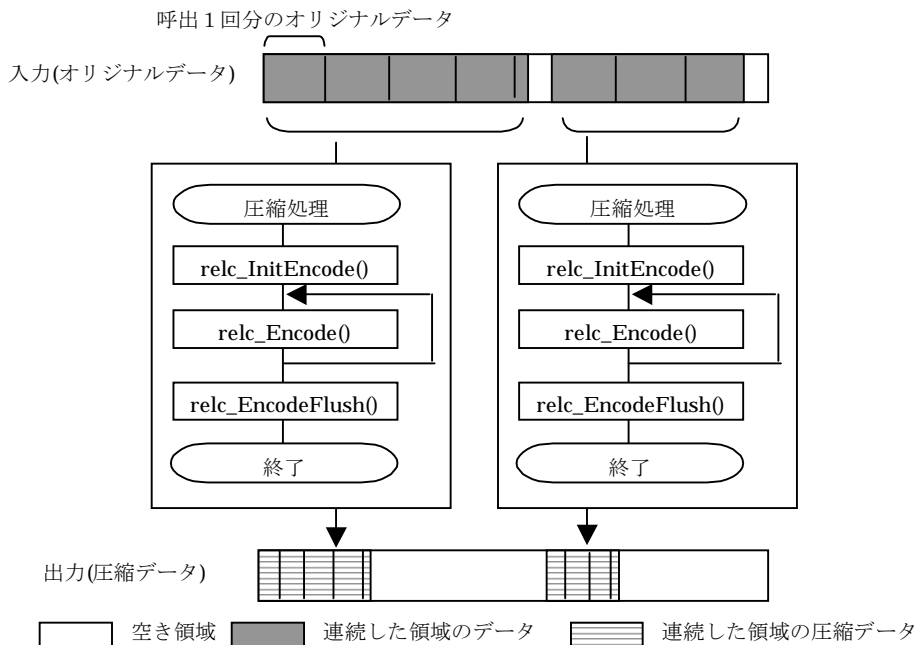
### [説明]

一連のオリジナルデータを **relc\_Encode( )** で処理する前に呼び出してください。引数として **nEncodeLevel** へ圧縮モードを指定する必要があります。

**pInfoArea** は低レベル用の制御情報領域へのポインタを指定します。

**nEncodeLevel** は圧縮モードを指定します。大きな数値ほど圧縮能力は高くなり、処理速度は低くなります。圧縮モードの指定方法については、「5 API」を参照してください。

圧縮処理は、次のようなイメージになります。



### 8.3 relc\_InitDecode ( )

[機能]

復元処理のための準備を行います。

[形式]

```
int relc_InitDecode(  
void RELC_FAR *      pInfoArea  
void  
)
```

[引数]

**pInfoArea**            低レベル API 用制御情報領域へのポインタ  
無し

[戻り値]

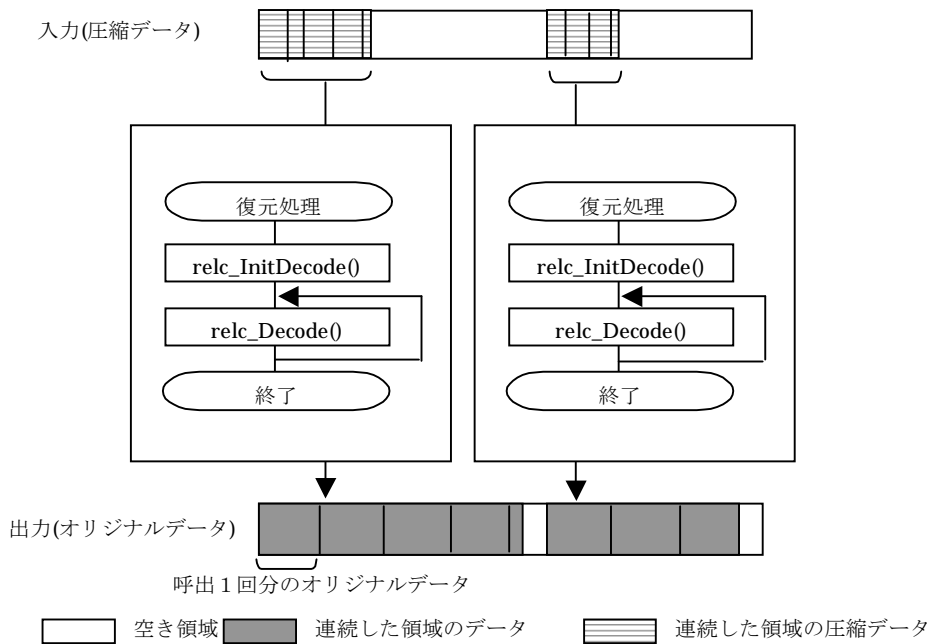
REL\_CV\_NORMAL    ==    正常終了

[説明]

連続した領域の圧縮データをrelc\_Decode()で処理する前に呼び出してください。

**pInfoArea** は低レベル用の制御情報領域へのポインタを指定します。

復元処理は、次のようなイメージになります。



## 8.4 relc\_Encode ( )

### [機能]

オリジナルデータを圧縮します。

### [形式]

```
int relc_Encode(
unsigned char RELC_FAR *    pInBuf ,
unsigned char RELC_FAR *    pOutBuf ,
unsigned short RELC_FAR *   pnInBuf ,
unsigned short RELC_FAR *   pnOutBuf ,
void RELC_FAR *            pInfoArea
)
```

### [引数]

<b>pInBuf</b>	オリジナルデータの入力バッファアドレス
<b>pOutBuf</b>	圧縮データの出力バッファアドレス
<b>pnInBuf</b>	[in ] 入力バッファ中のオリジナルデータのバイト数 [out] 入力バッファ中で使用したオリジナルデータのバイト数
<b>pnOutBuf</b>	[in ] 無視 [out] 出力バッファへ出力された圧縮データのバイト数
<b>pInfoArea</b>	低レベル API 用制御情報領域へのポインタ

### [戻り値]

<b>RELC_RV_NORMAL</b>	==	正常終了
<b>RELC_EV_ILLPARAM</b>	==	入力サイズエラー

### [説明]

**pInBuf** から **pnInBuf** バイトのオリジナルデータを読み取り、圧縮データを **pOutBuf** へ書き込みます。**pnInBuf** へは、最大 **RELC\_MAX\_ORIGINAL(32768)** が指定できます。関数から復帰すると、**pnInBuf** へ使用されたオリジナルデータのバイト数、**pnOutBuf** へ出力された圧縮データのバイト数が格納されています。

**pInfoArea** は低レベル用の制御情報領域へのポインタを指定します。

**RELC\_MAX\_ORIGINAL** バイト以上のオリジナルデータを圧縮する場合は、ループ処理を行い **RELC\_MAX\_ORIGINAL** 毎にこの関数を呼び出してください。関数を呼び出した後に得られる **pnInBuf** を減算することで、残りのオリジナルデータのバイト数が求まります。残りのオリジナルデータサイズが **RELC\_REUSE\_SIZE** 以下になったらループを終了し、**relc\_EncodeFlush()** を呼び出してください。

**pInBuf** の領域への操作は読み取りのみで書き込みは行いません。**pOutBuf** の領域への操作は書き込みのみで読み取りは行いません。

## 8.5 relc\_EncodeFlush ( )

### [機能]

圧縮の処理を終了します。

### [形式]

```
int relc_EncodeFlush (  
  unsigned char RELC_FAR *   pInBuf ,  
  unsigned char RELC_FAR *   pOutBuf ,  
  unsigned short RELC_FAR *  pnInBuf ,  
  unsigned short RELC_FAR *  pnOutBuf ,  
  void RELC_FAR *           pInfoArea  
)
```

### [引数]

<b>pInBuf</b>	オリジナルデータの入力バッファアドレス
<b>pOutBuf</b>	圧縮データの出力バッファアドレス
<b>pnInBuf</b>	[in ] 入力バッファ中のオリジナルデータのバイト数 [out] 入力バッファへ使用したバイト数
<b>pnOutBuf</b>	[in ] 無視 [out] 出力バッファへ出力したバイト数
<b>pInfoArea</b>	低レベル API 用制御情報領域へのポインタ

### [戻り値]

<b>RELC_RV_NORMAL</b>	==	正常終了
<b>RELC_EV_ILLDATA</b>	==	呼び出し順序不正エラー

### [説明]

**pInBuf** から圧縮データを読み取り、終端を含めた圧縮データを **pOutBuf** へ書き込みます。関数から復帰すると、**pnInBuf** へ使用されたオリジナルデータのバイト数、**pnOutBuf** へ出力した圧縮データのバイト数が格納されています。

**pInfoArea** は低レベル用の制御情報領域へのポインタを指定します。

この呼び出しにより、圧縮データの末尾となる終端コードが出力されます。

## 8.6 relc\_Decode ( )

### [機能]

圧縮データを復元します。

### [形式]

```
int relc_Decode(  
  unsigned char RELC_FAR *   pInBuf ,  
  unsigned char RELC_FAR *   pOutBuf ,  
  unsigned short RELC_FAR *  pnInBuf ,  
  unsigned short RELC_FAR *  pnOutBuf ,  
  void RELC_FAR *           pInfoArea  
)
```

### [引数]

<b>pInBuf</b>	圧縮データの入力バッファアドレス
<b>pOutBuf</b>	オリジナルデータの出力バッファアドレス
<b>pnInBuf</b>	[in ] 無視 [out] 入力バッファへ使用したバイト数
<b>pnOutBuf</b>	[in ] 無視 [out] 出力バッファへ出力したバイト数
<b>pInfoArea</b>	低レベル API 用制御情報領域へのポインタ

### [戻り値]

RELC_RV_NORMAL	==	正常終了
RELC_RV_ENDEDATA	==	全ての圧縮データを処理した
RELC_EV_ILLDATA	==	データエラー

### [説明]

**pInBuf** から圧縮データを読み取り、復元したオリジナルデータを **pOutBuf** へ書き込みます。関数から復帰すると、**pnInBuf** へ圧縮データへのバッファアドレスの更新バイト数、**pnOutBuf** へオリジナルデータデータへのバッファアドレスの更新バイト数が格納されています。

**pInfoArea** は低レベル用の制御情報領域へのポインタを指定します。

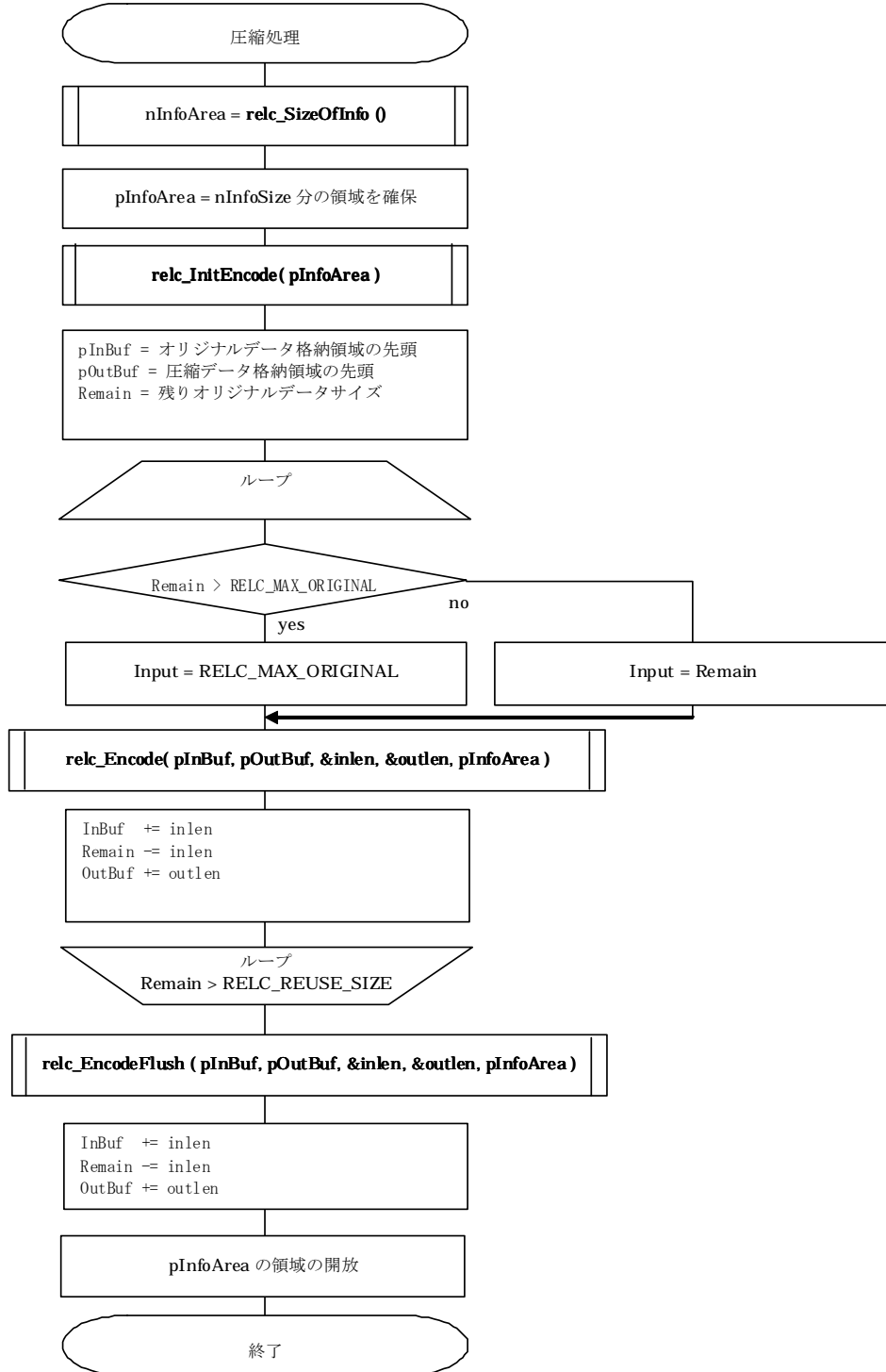
全ての圧縮データの復元を行うと戻り値として **RELC\_RV\_ENDEDATA** を返します。

**pInBuf** の領域への操作は読み取りのみで書き込みは行いません。**pOutBuf** の領域への操作は読み書きを行います。

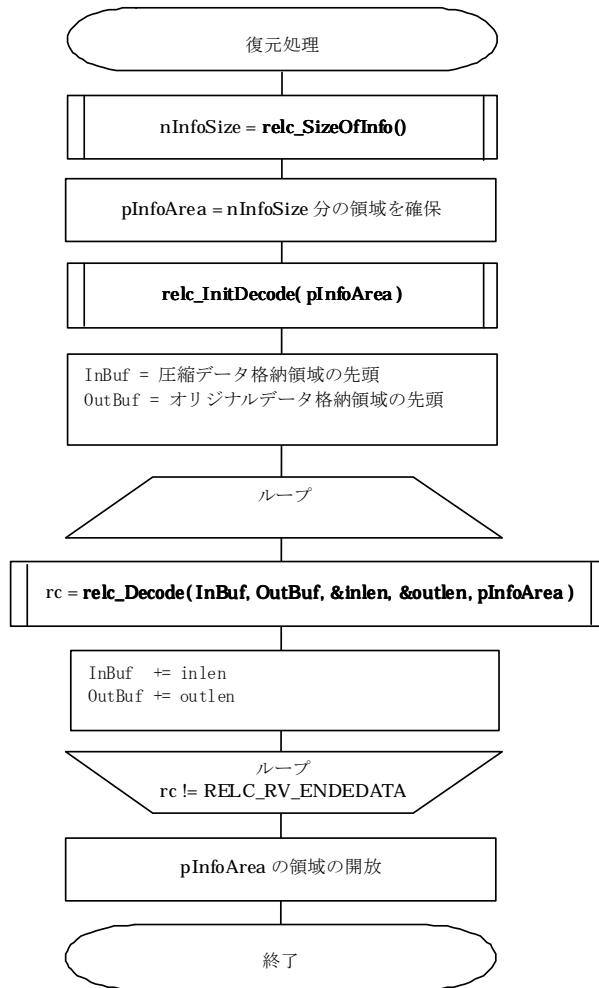
この関数では、処理速度を優先するため厳密なエラーチェックは行いません。従って、圧縮データの正当性はユーザー側で保障してください。

## 8.7 低レベル API の使用方法

### 8.7.1 圧縮処理



## 8.7.2 復元処理



## 9 定義

本製品の定義について説明します。定義は、`relc_mt.h`, `relcm_mt.h`, `relch_mt.h`, `relctype.h` で記述されています。

### 9.1 定数

マクロ	値 <sup>(10)</sup>	説明
<code>RELC_MAX_ORIGINAL</code>	32768	呼び出し1回毎の最大オリジナルデータサイズ
<code>RELC_MAX_CODE</code>	32770	呼び出し1回毎の最大圧縮データサイズ
<code>RELC_REUSE_SIZE</code>	8192	バッファリング時の再利用サイズおよび圧縮時の終了条件
<code>RELC_MEV_MASK</code>	255	中レベルAPI用のエラー取得マスク値
<code>RELC_MODE_MAXIMUM</code>	0	最高圧縮モード
<code>RELC_MODE_HIGH</code>	32	高圧縮モード
<code>RELC_MODE_MIDDLE</code>	16	中圧縮モード
<code>RELC_MODE_LOW</code>	4	低圧縮モード
<code>RELC_MODE_MINIMUM</code>	1	最低圧縮モード
<code>RELC_KIND_NO_INFO</code>	1	ヘッダー情報無し
<code>RELC_KIND_ZERO</code>	2	付加情報を付加しない
<code>RELC_KIND_ORGSIZE</code>	4	オリジナルサイズ付加
<code>RELC_KIND_CODESIZE</code>	8	圧縮サイズ付加
<code>RELC_RV_NORMAL</code>	0	正常終了時の戻り値
<code>RELC_RV_ENDEDATA</code>	1	復元終了時の戻り値
<code>RELC_RV_CONTINUE</code>	2	関数を呼び出しの継続要求
<code>RELC_EV_ILLPARAM</code>	10	パラメータ異常
<code>RELC_EV_ILLDATA</code>	11	異常データ検出
<code>RELC_EV_ILLFORM</code>	12	フォーマットが異常
<code>RELC_EV_STOPUSER</code>	13	要求による中断
<code>RELC_EV_CANTOPEN</code>	20	ファイルオープンエラー
<code>RELC_EV_FILEREAD</code>	21	ファイル読み取りエラー
<code>RELC_EV_FILEWRITE</code>	22	ファイル書き込みエラー
<code>RELC_EV_CLOSE</code>	23	ファイルのクローズエラー
<code>RELC_EV_NODATA</code>	30	データ不足
<code>RELC_EV_NOBUFF</code>	31	バッファ不足
<code>RELC_MCV_NODATA</code>	256	入力データが無い
<code>RELC_MCV_NOBUFF</code>	512	出力バッファが無い
<code>RELC_MCV_EDTEND</code>	1024	全処理を終了

### 9.2 型

型名	説明
<code>RELC_64VAR</code>	64ビットの符号付き整数型
<code>RELCHTYPE</code>	高レベルAPI用制御情報型
<code>RELCMTYPE</code>	中レベルAPI用制御情報型
<code>RELCTYPE</code>	低レベルAPI用制御情報型

## 10 エラーリファレンス

エラーコードについて説明します。

定義	説明
RELC_RV_NORMAL	正常終了しました。
RELC_RV_ENDEDATA	全ての圧縮データを処理しました。
RELC_RV_CONTINUE	継続して API を呼び出してください。
RELC_EV_ILLPARAM	パラメータが異常です。
RELC_EV_ILLDATA	圧縮データが異常です。またはフラッシュ動作の呼び出しタイミングが違います。
RELC_EV_ILLFORM	フォーマットが異常です。
RELC_EV_STOPUSER	要求により処理を中断しました。
RELC_EV_CANTOPEN	ファイルオープンに失敗しました。
RELC_EV_FILEREAD	ファイル読み取りに失敗しました。
RELC_EV_CLOSE	ファイルのクローズでエラーがありました。
RELC_EV_FILEWRITE	ファイル書き込みに失敗しました。

## 11 PC 版評価プログラムについて

本製品には、PC 上で圧縮ファイルを作成、および、作成された圧縮ファイルを復元するための評価プログラム (RelcEva.exe) が添付されています。このプログラムは、評価用のみ利用できます。

(このプログラムを配布することはできません。システムに利用する場合は、別途ライセンス製品を購入する必要があります。)

### ・ 起動方法

prompt\$ RelcEva.exe option infile outfile [mode]

option 書式 {E [{N | OC}] | D [N]}<sup>1</sup>

**圧縮** 圧縮処理の場合は、'E' を指定します。  
圧縮形式を指定する場合は、'E' に続けてオプション 1 を記述します。

オプション 1	説明
N	RELC_KIND_NO_INFO 形式の圧縮ファイルを作成します。
O	RELC_KIND_ORGSIZE 形式の圧縮ファイルを作成します。
C	RELC_KIND_CODESIZE 形式の圧縮ファイルを作成します。

オプション 1 を省略した場合は、KIND\_ZERO 形式の圧縮ファイルを作成します。

'O' と 'C' は併記することも可能です。その場合、RELC\_KIND\_ORGSIZE | RELC\_KIND\_CODESIZE 形式の圧縮ファイルを作成します。

**復元** 復元処理の場合は、'D' を指定します。  
圧縮形式を指定する場合は、'D' に続けてオプション 2 を記述します。

オプション 2	説明
N	RELC_KIND_NO_INFO 形式の圧縮ファイルを復元します。

オプション 2 を省略した場合は、RELC\_KIND\_NO\_INFO 形式以外の圧縮ファイルを復元します。

infile 入力ファイル名を指定します。

outfile 出力ファイル名を指定します。

mode 圧縮モードを指定します。(0 または 1 ~128 )  
指定の際の目安として本製品の定数値を示します。


値	定数
0	RELC_MODE_MAXIMUM
32	RELC_MODE_HIGH
16	RELC_MODE_MIDDLE
4	RELC_MODE_LOW
1	RELC_MODE_MINIMUM

※圧縮時のみ有効で、復元時は無視されます。

<sup>1</sup> 大文字小文字の区別はありません。また、記号の意味は次のとおりです。

[...] は、その中の要素は記述が省略可能であることを示します

{.|.} は、その中の要素内で一つを選択することを示します

  
**FUJITSU**