

データ圧縮ライブラリ

ESLC for Windows

ユーザーズマニュアル

02 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

ESLC for Windows 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。Microsoft、Windows、Windows Server、Visual Studio、Visual C++ は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

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。

■ 製品名称

本マニュアルでは、製品名称を次のように略して表記しています。

Windows Server™ 2003 operating system を Windows Server 2003 と表記しています。

Windows® 2000 Server operating system を Windows 2000 Server と表記しています。

Windows Vista™ operating system を Windows Vista と表記しています。

Windows® XP operating system を Windows XP と表記しています。

Windows® 2000 operating system を Windows 2000 と表記しています。

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ありません。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。

目次

1	概要.....	1
2	動作環境.....	1
3	開発環境.....	1
4	組み込み方法	2
4.1	ヘッダーファイルのインクルード.....	2
4.2	コンパイル時のシンボル定義.....	2
4.3	ライブラリのリンク	3
4.4	呼び出し規約の整合のための宣言.....	3
4.5	入出力バッファ	4
4.6	圧縮 / 復元制御用テーブル (SLC1_ELEM)	5
5	圧縮データの構造	6
6	定義.....	7
6.1	定数	7
6.2	構造体.....	8
6.3	関数	8
7	関数.....	9
7.1	slc1_init_multi ().....	10
7.2	slc1_exit ().....	11
7.3	slc1_flg_check ()	12
7.4	slc1_flg_reset ().....	13
7.5	slc1_encode ().....	14
7.6	slc1_decode ().....	15
7.7	slc1_file_encode ().....	16
7.8	slc1_file_decode ().....	17
8	使用例	18
8.1	圧縮処理	18
8.2	復元処理	21
8.3	連続して複数ファイルを処理する場合.....	24
9	エラーリファレンス	25
10	PC 版評価プログラムについて.....	26

1 概要

本製品は、ファイルやメモリに格納されたデータに対し圧縮および復元を行うためのライブラリ製品です。アルゴリズムは、富士通研究所が開発した損失の無い（ロスレス）圧縮方式を採用しています。

本製品には、次のライブラリとヘッダーファイルが同封されています。

本製品ライブラリ名	本製品との結合方式	C ライブラリとの結合方式	備考
eslc_static_mt.lib	static	static	圧縮・復元両用
eslc_static_md.lib	static	dynamic	圧縮・復元両用
eslc_dynamic_mt.dll eslc_dynamic_mt.lib	dynamic	static	圧縮・復元両用 DLL インポートライブラリ
eslc_dynamic_md.dll eslc_dynamic_md.lib	dynamic	dynamic	圧縮・復元両用 DLL インポートライブラリ
eslc_mult.h	-	-	C 言語用ヘッダーファイル

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。

2 動作環境

本製品の動作環境は、次のとおりです。

OS : Windows Server 2003
Windows 2000 Server
Windows Vista
Windows XP
Windows 2000

動作環境（OS）は、すべて日本語版で、x86（32 ビット）ベースのコンピュータ上で動作するものです。

3 開発環境

本製品は、Microsoft 社製の開発ツールである Visual Studio 2005 に含まれる Visual C++ 2005 を用いてコンパイルしてあります。ライブラリ使用時には、 Visual C++ 2005 を使用してください。

4 組み込み方法

ライブラリの組み込み方法について説明します。

4.1 ヘッダーファイルのインクルード

ライブラリを使用するソースファイルで“eslc_mult.h”をインクルードしてください。

4.2 コンパイル時のシンボル定義

コンパイル時には、次のシンボルを定義してください。

- スタティックリンクライブラリ (eslc_static_*.lib) を使用する場合
“_ANSI”, “SERVER”
- ダイナミックリンクライブラリ (eslc_dynamic_*.dll) を使用する場合
“_ANSI”, “SERVER”, “_USRDLL”

4.3 ライブラリのリンク

用途に応じたライブラリを選択し、作成されたプログラムとリンクしてください。

コンパイルオプションに “/MD”, “/MT” をそれぞれ指定して作成した 2 種類のライブラリがあります。コンパイルオプションの詳細については、Microsoft 社の MSDN ライブラリ等を参照ください。

実行ファイルと DLL のリンクには 2 つの方法があります。

- DLL インポートライブラリ (eslc_dynamic_*.lib) を使用する
使用するアプリケーションに dll インポートライブラリ (eslc_dynamic_*.lib) をリンクします。この方法は、DLL を使用する手続きが自動化されるために簡単に DLL (eslc_dynamic_*.dll) を利用できます。Microsoft Visual C++ では、「暗黙的リンク」と説明されている方法です。

Visual C++ 2005 以外をご使用の場合は、DLL インポートライブラリが使用できません。

- ダイナミックリンクライブラリ (eslc_dynamic_*.dll) をロードする
実行時に Windows API を使用してダイナミックリンクライブラリ (eslc_dynamic_*.dll) をロードし関数アドレスを取得します。Microsoft Visual C++ では、「明示的リンク」と説明されている方法です。

4.4 呼び出し規約の整合のための宣言

本製品には、スタティックライブラリとダイナミックリンクライブラリが含まれています。ダイナミックリンクライブラリにおいては、他言語との呼び出し規約を合わせるために “WINAPI” を付加して関数定義をしています。ダイナミックリンクライブラリをご使用の場合は、コンパイルスイッチに “_USRDLL” を付加してください。

スタティックライブラリは、通常の C 呼び出し規約で関数を定義しています。

4.5 入出力バッファ

処理を行うために入出力バッファを獲得する必要があります。

圧縮データは、2 バイトのヘッダーを付加します。従って、圧縮処理の場合は出力バッファ、復元処理の場合は入力バッファのバッファサイズを 2 バイト余分に確保してください。

次にバッファサイズを本製品処理可能な最大値(SLC1_BUF_SIZE)にする場合の入出力バッファサイズの関係を示します。

- ・圧縮時：入力バッファサイズ = SLC1_BUF_SIZE
出力バッファサイズ = SLC1_BUF_SIZE + 2
- ・復元時：入力バッファサイズ = SLC1_BUF_SIZE + 2
出力バッファサイズ = SLC1_BUF_SIZE

本製品で処理可能なバッファサイズは1024 ~ 32768バイトです。ただし、圧縮時の出力バッファ、復元時の入力バッファはこの値+2 バイトの範囲となります。

SLC1_BUF_SIZE は、入出力バッファ獲得時の推奨値（最大値）の 32768 バイトです。バッファサイズを小さくするほど圧縮率が低下しますので、SLC1_BUF_SIZE を使用することをお奨めします。

4.6 圧縮 / 復元制御用テーブル (SLC1_ELEM)

マルチスレッド処理を行うために、`slc1_init_multi ()` 内で圧縮 / 復元制御用テーブル (SLC1_ELEM) 管理領域を獲得します。

`slc1_init_multi ()` が正常終了した場合、戻り値としてこの領域へのポインタ (SLC1_ELEM *) が戻ってきますので、通常はこれを `auto` 変数に格納し、以降、各関数呼び出し時には、この変数を引数として呼び出します。なお、圧縮 / 復元制御用テーブルは、内部処理で使用しますので変更しないようにしてください。

この領域は、`slc1_exit ()` を呼び出すことで開放されます。

[例]

```
{
  SLC1_ELEM *Thread1;    /* スレッド 1 */
  SLC1_ELEM *Thread2;    /* スレッド 2 */
  :
  Thread1 = slc1_init_multi (SLC1_BUF_SIZE, 7); /* 領域獲得 */
  :
  ( 圧縮 / 復元処理 )
  :
  slc1_exit (Thread1);    /* 領域開放 */
}
```

1 スレッドにつき 1 つの圧縮 / 復元制御用テーブル管理領域を用意する必要があり、別スレッドの圧縮 / 復元制御用テーブルを使用した場合、正常に圧縮 / 復元処理を行うことができなくなります。

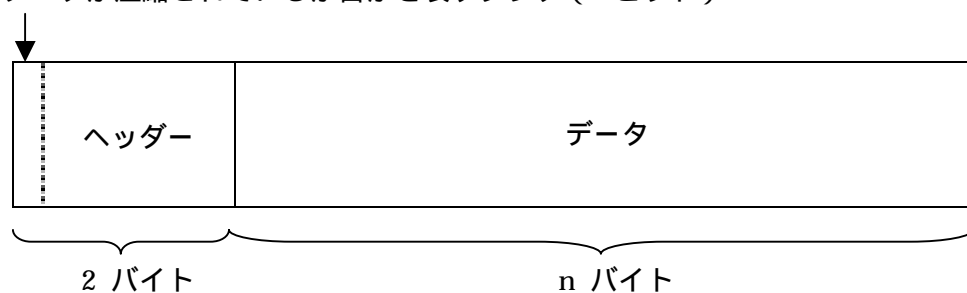
1 スレッドで複数のファイルを処理する場合は、「8.3 連続して複数ファイルを処理する場合」を参照してください。

5 圧縮データの構造

圧縮データの構造を以下に示します。

- ・ヘッダー : データが圧縮されているか否か、データが何バイトあるかの情報
- ・データ : ヘッダーが示すバイト数のデータ

データが圧縮されているか否かを表すフラグ (1 ビット)



(データのサイズ n バイト : ビッグエンディアン)

図1 圧縮データの構造

6 定義

本製品の定義について説明します。

これらは、eslc_mult.hで定義しています。

6.1 定数

NORMAL	0
BADCODE	1
ERR_NOMEM	2
NOBUSY	0
BUSY	1
ERR_INFILE_OPEN	21
ERR_OUTFILE_OPEN	22
ERR_FILE_READ	23
ERR_FILE_WRITE	24
ERR_INFILE_CLOSE	25
ERR_OUTFILE_CLOSE	26
SLC1_BUF_SIZE	(unsigned int) ((unsigned int) 8192*4) 入出力バッファ獲得時の推奨値です。
SLCDECLARE	WINAPI [“_USRDLL”を定義している場合] “_USRDLL”を定義していない場合、 ”SLCDECLARE”は値なしで定義されます。

6.2 構造体

構造体名	説明
E_NODE	圧縮用情報テーブルです。
D_NODE	復元用情報テーブルです。
SLC1_ELEM	圧縮 / 復元制御用テーブルです。

6.3 関数

関数名	説明
slc1_init_multi()	メモリ領域の確保 / 圧縮レベルの設定を行います。
slc1_exit()	確保したメモリ領域を開放し終了処理を行います。
slc1_flg_check()	BUSY フラグのチェックを行います。
slc1_flg_reset()	BUSY フラグのリセットを行います。
slc1_encode()	データを圧縮します。
slc1_decode()	圧縮データを復元します。
slc1_file_encode()	1 ファイルを圧縮します。
slc1_file_decode()	1 ファイルを復元します。

7 関数

本製品では、次の 8 つの関数を提供しています。

`slc1_init_multi()`

`slc1_exit()`

`slc1_flg_check()`

`slc1_flg_reset()`

`slc1_encode()`

`slc1_decode()`

`slc1_file_encode()`

`slc1_file_decode()`

以降の説明では、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。

7.1 slc1_init_multi ()

[機能]

メモリ領域の確保 / 圧縮レベルの設定を行います。

[形式]

```
SLC1_ELEM * SLCDECLARE slc1_init_multi (  
unsigned short      buff_size,  
int                level  
)
```

[引数]

buff_size 出力用としてアクセス可能な領域サイズ
level 圧縮レベル (0~7)

[戻り値]

SLC1_ELEM *	!=	NULL	正常終了
	==	NULL	領域の確保失敗

[説明]

必ず各スレッドの最初に呼び出してください。

本関数が正常終了した場合、戻り値として圧縮 / 復元制御用テーブルへのポインタが戻ります。このポインタは、以降の各関数の引数に使用します。

本関数を呼び出すと、必ず `slc1_exit ()` で終了する必要があります。

buff_size へは、ライブラリがアクセス可能な領域サイズを指定します。
この値は、次の関数の処理で利用します。

・ `slc1_decode()`

異常なデータ (正常に圧縮されたものでないものや破壊されたもの) を復元させようとした場合に、予期せぬ大きさの書き込み処理を行う可能性があります。**buff_size** は、このような現象による領域破壊を防ぐために利用されます。

圧縮時に用いた領域サイズと同じ値を指定するようにしてください。

・ `slc1_file_encode()`

与えられた入出力バッファへのポインタから入力バッファが **buff_size** バイト、出力バッファが **buff_size+2** バイトあるものとしてアクセスします。

・ `slc1_file_decode()`

与えられた入出力バッファへのポインタから入力バッファが **buff_size+2** バイト、出力バッファが **buff_size** バイトあるものとしてアクセスします。`slc1_decode()`と同様に異常なデータが入力された場合の領域破壊防止にも使用します。

圧縮時に用いた領域サイズと同じ値を指定するようにしてください。

level へは、圧縮レベルを指定します。

圧縮レベルは0~7の範囲で指定することができ、範囲外の値を指定した場合はデフォルト値である7に設定します。圧縮レベルが大きいほど圧縮率は高くなりますが、必要となるRAM領域が大きくなります。

なお、圧縮と復元では同じ圧縮レベルを指定する必要があります。

7.2 slc1_exit ()

[機能]

確保したメモリ領域を開放し終了処理を行います。

[形式]

```
void SLCDECLARE slc1_exit (  
SLC1_ELEM          *blsadr  
)
```

[引数]

blsadr 圧縮 / 復元制御用テーブルへのポインタ

[戻り値]

なし

[説明]

必ず各スレッドの最後に呼び出してください。

7.3 slc1_flg_check()

[機能]

BUSY フラグのチェックを行います。

[形式]

```
int SLCDECLARE slc1_flg_check(  
SLC1_ELEM          *blsadr  
)
```

[引数]

blsadr 圧縮 / 復元制御用テーブルへのポインタ

[戻り値]

int	==	NOBUSY	圧縮 / 復元処理実行可能状態
	==	BUSY	圧縮 / 復元処理実行不可能状態

[説明]

必ず一連の連続したデータ処理を開始する前に呼び出してください。

7.4 slc1_flg_reset ()

[機能]

BUSY フラグのリセットを行います。

[形式]

```
void SLCDECLARE slc1_flg_reset (
SLC1_ELEM          *blsadr
)
```

[引数]

blsadr 圧縮 / 復元制御用テーブルへのポインタ

[戻り値]

なし

[説明]

必ず一連の連続したデータ処理が終了した後に呼び出してください。

7.5 slc1_encode ()

[機能]

データを圧縮します。

[形式]

```
int SLCDECLARE slc1_encode (  
  unsigned char      *InBuf,  
  unsigned char      *OutBuf,  
  unsigned short int  inlen,  
  unsigned short int  *outlen,  
  SLC1_ELEM          *blsadr  
)
```

[引数]

InBuf	オリジナルデータを格納した入力バッファへのポインタ
OutBuf	圧縮データを格納する出力バッファへのポインタ
inlen	入力バッファ中のオリジナルデータのバイト数 (入力バッファサイズ以下の値)
outlen	[in] 無視
blsadr	[out] 出力バッファに格納した圧縮データのバイト数 圧縮 / 復元制御用テーブルへのポインタ

[戻り値]

int	==	NORMAL	正常終了
	==	ERR_NOMEM	メモリエラー
	==	BADCODE	inlen = 0

[説明]

InBuf から inlen バイトのオリジナルデータを読み取り、圧縮したデータを OutBuf へ書き込みます。

圧縮データのバイト数は、outlen へ書き込みます。

inlen には、1 以上の値を設定してください。

7.6 slc1_decode ()

[機能]

圧縮データを復元します。

[形式]

```
int SLCDECLARE slc1_decode (  
unsigned char      *InBuf,  
unsigned char      *OutBuf,  
unsigned short int *inlen,  
unsigned short int *outlen,  
SLC1_ELEM         *blsadr  
)
```

[引数]

InBuf	圧縮データを格納した入力バッファへのポインタ
OutBuf	オリジナルデータを格納する出力バッファへのポインタ
inlen	[in] 無視
	[out] 実際に処理した圧縮データのバイト数
outlen	[in] 無視
	[out] 出力バッファに格納したオリジナルデータのバイト数
blsadr	圧縮 / 復元制御用テーブルへのポインタ

[戻り値]

int	==	NORMAL	正常終了
	==	BADCODE	復元できないデータ

[説明]

InBuf から圧縮データを読み取り、復元したデータ (オリジナルデータ) を **OutBuf** へ書き込みます。

実際に処理した圧縮データのバイト数は、**inlen** へ書き込みます。

オリジナルデータのバイト数は、**outlen** へ書き込みます。

7.7 slc1_file_encode ()

[機能]

1 ファイルを圧縮します。

[形式]

```
int SLCDECLARE slc1_file_encode (
unsigned char      *InBuf,
unsigned char      *OutBuf,
char              *InFname,
char              *OutFname,
SLC1_ELEM         *blsadr
)
```

[引数]

InBuf	入力バッファへのポインタ
OutBuf	出力バッファへのポインタ
InFname	入力ファイル名へのポインタ
OutFname	出力ファイル名へのポインタ
blsadr	圧縮 / 復元制御用テーブルへのポインタ

[戻り値]

int	==	NORMAL	正常終了
	==	ERR_NOMEM	メモリエラー
	==	ERR_INFILE_OPEN	入力ファイルオープンエラー
	==	ERR_OUTFILE_OPEN	出力ファイルオープンエラー
	==	ERR_FILE_READ	ファイルリードエラー
	==	ERR_FILE_WRITE	ファイルライトエラー
	==	ERR_INFILE_CLOSE	入力ファイルクローズエラー
	==	ERR_OUTFILE_CLOSE	出力ファイルクローズエラー

[説明]

入力ファイルからオリジナルデータを読み取り、圧縮したデータを出力ファイルへ書き込みます。

本関数は、「7.1 slc1_init_multi()」で説明した **buff_size** 分のデータを入力ファイルから読み取りslc1_encode()を呼び出す処理を入力ファイルのデータを読み終わるまで繰り返します。

InBuf, **OutBuf** で与えられた入出力バッファ領域は内部処理で使用します。本関数では、与えられた入出力バッファへのポインタから入力バッファが **buff_size** バイト、出力バッファが **buff_size+2** バイトあるものとしてアクセスしますので、本関数呼び出し前に必要な入出力バッファ領域が確保されている必要があります。

7.8 slc1_file_decode ()

[機能]

1 ファイルを復元します。

[形式]

```
int SLCDECLARE slc1_file_decode(  
unsigned char      *InBuf,  
unsigned char      *OutBuf,  
char               *InFname,  
char               *OutFname,  
SLC1_ELEM          *blsadr  
)
```

[引数]

InBuf	入力バッファへのポインタ
OutBuf	出力バッファへのポインタ
InFname	入力ファイル名へのポインタ
OutFname	出力ファイル名へのポインタ
blsadr	圧縮 / 復元制御用テーブルへのポインタ

[戻り値]

int	==	NORMAL	正常終了
	==	BADCODE	復元できないデータ
	==	ERR_INFILE_OPEN	入力ファイルオープンエラー
	==	ERR_OUTFILE_OPEN	出力ファイルオープンエラー
	==	ERR_FILE_READ	ファイルリードエラー
	==	ERR_FILE_WRITE	ファイルライトエラー
	==	ERR_INFILE_CLOSE	入力ファイルクローズエラー
	==	ERR_OUTFILE_CLOSE	出力ファイルクローズエラー

[説明]

入力ファイルから圧縮データを読み取り、復元したデータ（オリジナルデータ）を出力ファイルへ書き込みます。

本関数は、一回の復元処理に必要なデータを入力ファイルから読み取りslc1_decode()を呼び出す処理を入力ファイルのデータを読み終わるまで繰り返します。

InBuf, **OutBuf** で与えられた入出力バッファ領域は内部処理で使用します。本関数では、与えられた入出力バッファへのポインタから入力バッファが「7.1 slc1_init_multi()」で説明した **buff_size** +2 バイト、出力バッファが **buff_size** バイトあるものとしてアクセスしますので、本関数呼び出し前に必要な入出力バッファ領域が確保されている必要があります。

8 使用例

本製品のインタフェースを使用して1ファイルの圧縮処理と復元処理の例を示します。
圧縮 / 復元処理は1ファイル毎に行い、EOF が現れるまで処理を続行します。

8.1 圧縮処理

圧縮処理フローを図 2、3 に示します。

1. 処理の開始時に `slc1_init_multi()` を呼び出します。
この時、ライブラリがアクセス可能な領域サイズ、圧縮レベルを設定します。
2. `slc1_flg_check()` を呼び出し、BUSY フラグをチェックします。
BUSY フラグが NOBUSY の場合、圧縮処理を行いません。
3. 圧縮処理を行いません。
`slc1_file_encode()` を使用する場合
`slc1_file_encode()` を呼び出し、入力ファイルのデータを圧縮して出力ファイルに書き込みます。

`slc1_encode()` を使用する場合
入力 / 出力ファイルをオープンした後、入力ファイルの EOF が現れるまで下記の処理を繰り返します。
 - ・圧縮するデータを入力ファイルから InBuf に読み込む
 - ・ `slc1_encode()` を呼び出す
 - ・ OutBuf のデータを出力ファイルへ書き込む1 ファイルを処理した後、入力 / 出力ファイルをクローズします。
4. `slc1_flg_reset()` を呼び出し、BUSY フラグをリセットします。
5. 最後に終了処理として `slc1_exit()` を呼び出します。

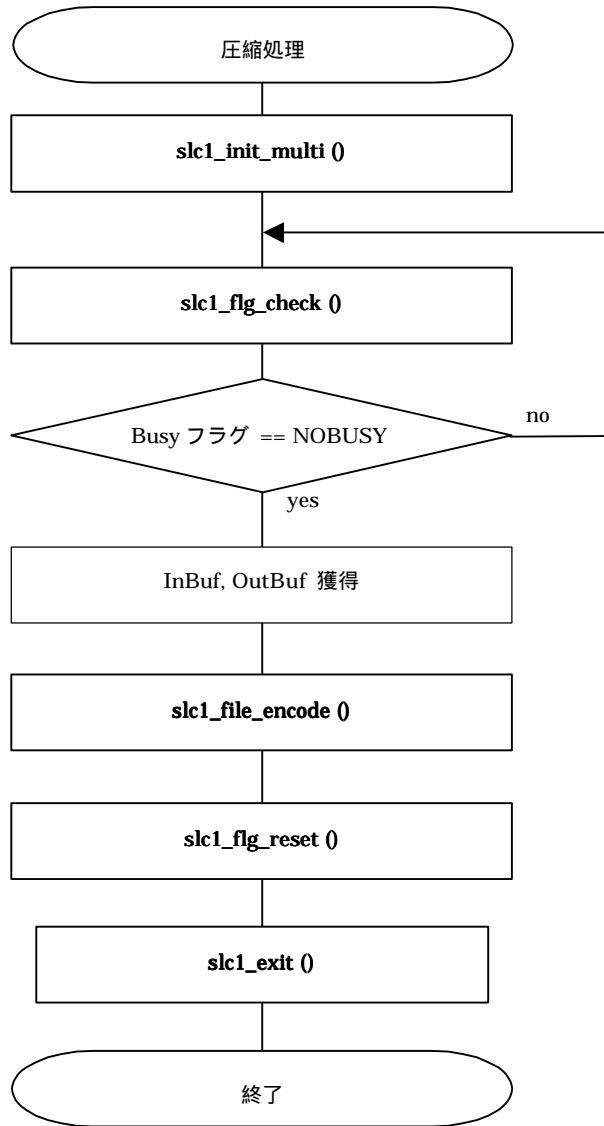
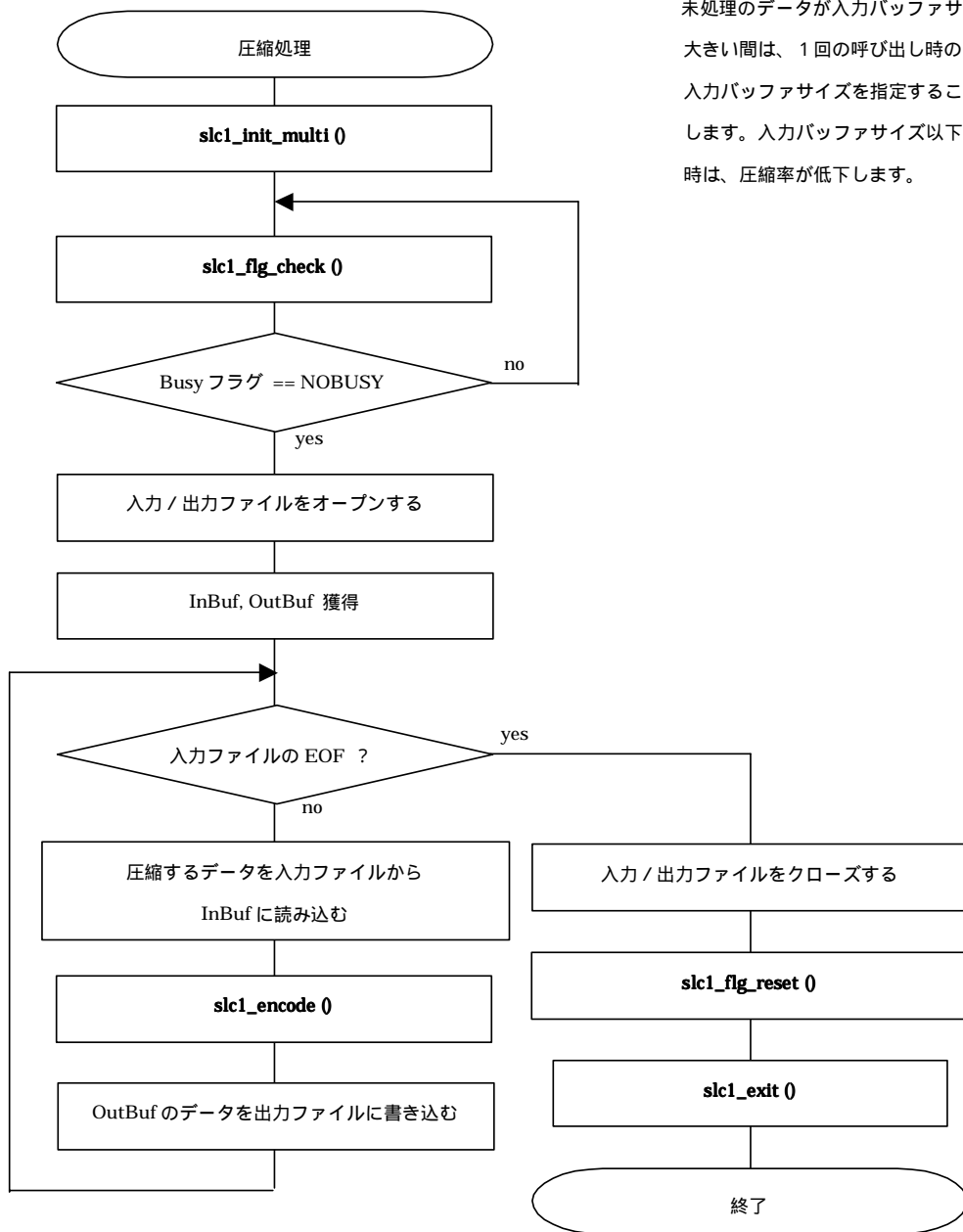


図 2 圧縮処理フロー (slc1_file_encode()を使用する場合)



未処理のデータが入力バッファサイズより大きい間は、1回の呼び出し時の inlen に入力バッファサイズを指定することを推奨します。入力バッファサイズ以下を渡した時は、圧縮率が低下します。

図 3 圧縮処理フロー (slc1_encode()を使用する場合)

8.2 復元処理

復元処理フローを図 4、5 に示します。

1. 処理の開始時に `slc1_init_multi()` を呼び出します。
この時、ライブラリがアクセス可能な領域サイズ、圧縮レベルを設定します。
2. `slc1_flg_check()` を呼び出し、BUSY フラグをチェックします。
BUSY フラグが NOBUSY の場合、復元処理を行ないます。
3. 復元処理を行ないます。
`slc1_file_decode()` を使用する場合
`slc1_file_decode()` を呼び出し、入力ファイルの圧縮データを復元して出力ファイルに書き込みます。

`slc1_decode()` を使用する場合
入力 / 出力ファイルをオープンした後、入力ファイルの EOF が現れるまで下記の処理を繰り返します。
 - ・入力ファイルからヘッダーを読み込み、バッファサイズを獲得する
 - ・バッファサイズ分だけ入力ファイルからデータを InBuf に読み込む
 - ・ `slc1_decode()` を呼び出す
 - ・ OutBuf のデータを出力ファイルへ書き込む1 ファイルを処理した後、入力 / 出力ファイルをクローズします。
4. `slc1_flg_reset()` を呼び出し、BUSY フラグをリセットします。
5. 最後に終了処理として `slc1_exit()` を呼び出します。

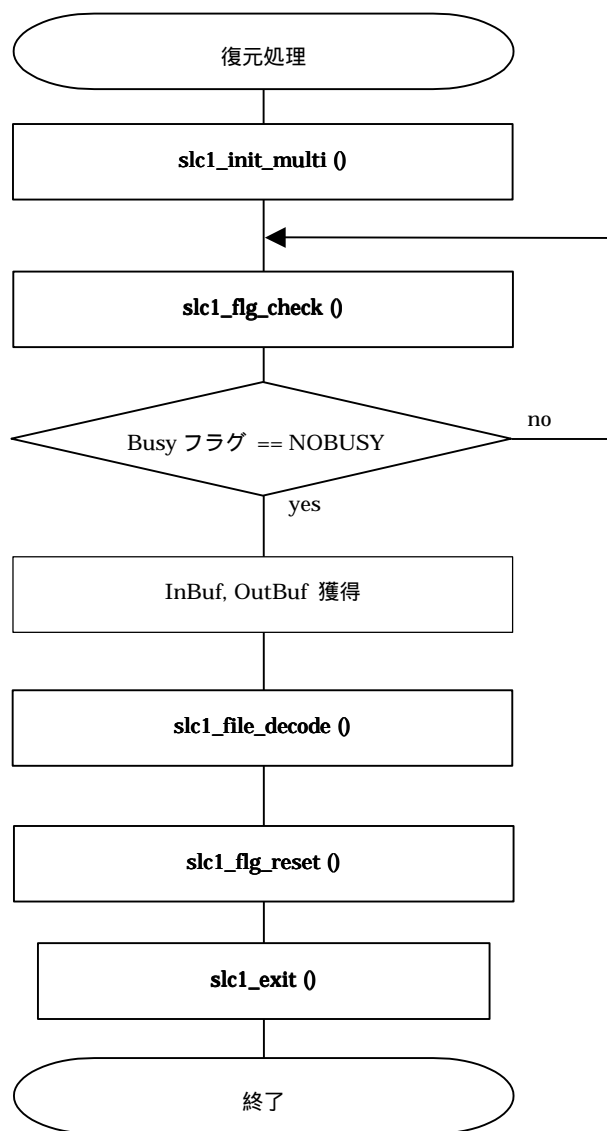


図 4 復元処理フロー (slc1_file_decode()を使用する場合)

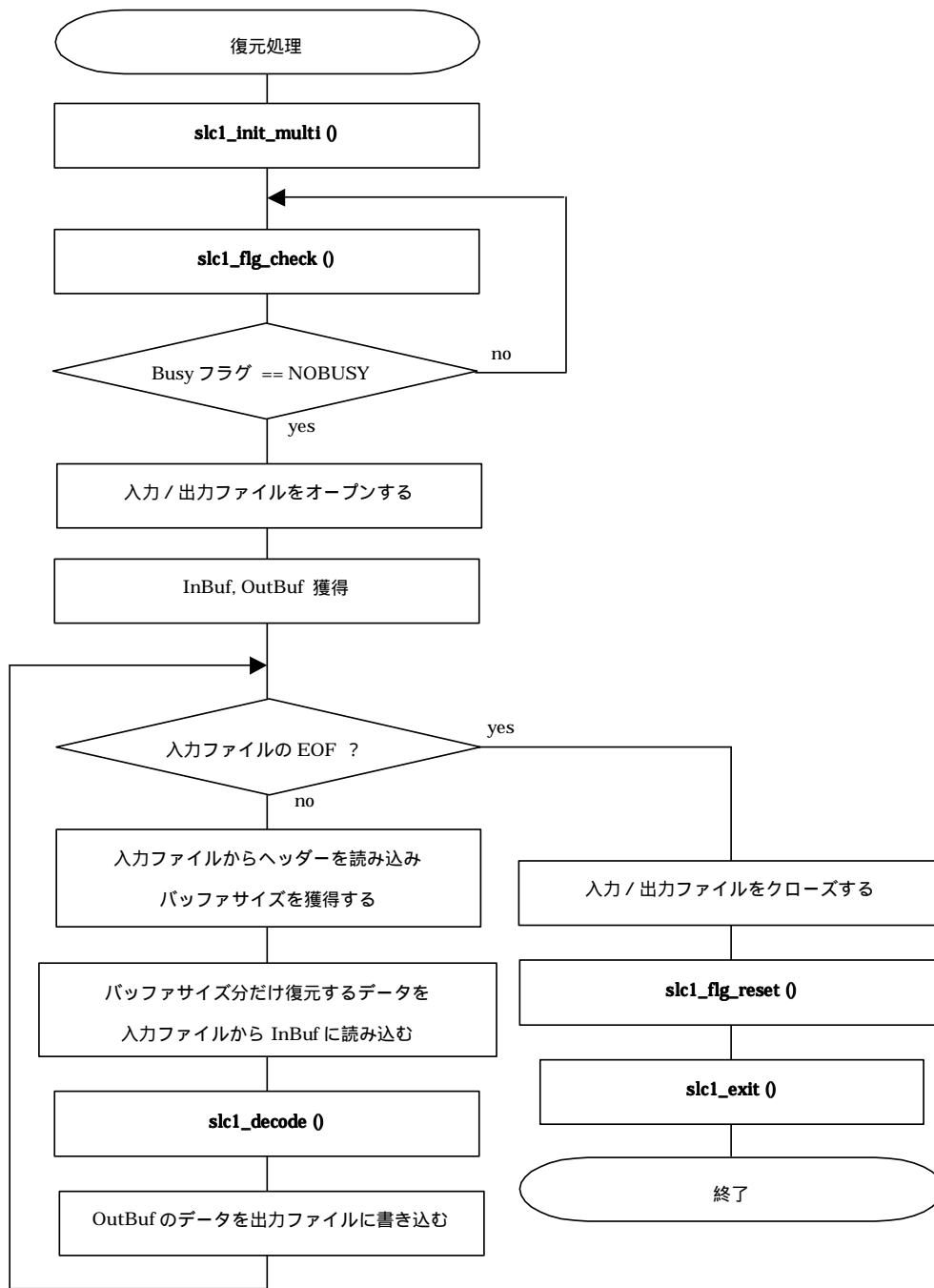


図 5 復元処理フロー (slc1_decode()を使用する場合)

8.3 連続して複数ファイルを処理する場合

各ファイルごとに圧縮レベルを変更する必要のない場合は、`slc1_init_multi()` を 1 スレッド起動時に 1 度だけ呼び出します。

1 ファイルの処理が完了し `slc1_flg_reset()` を呼び出した後に他のファイルの処理を行うことが可能です。

圧縮処理の後に復元処理を行うことも可能です。

全ての処理を終了する場合は、必ず `slc1_exit()` を呼び出してください。

また、圧縮レベルを変更したい場合は、`slc1_exit()` で一度処理を終了し、その後、再度 `slc1_init_multi()` を呼び出して圧縮レベルの再設定を行ってください。この場合も処理を終了する際には必ず `slc1_exit()` を呼び出してください。

1 度 `slc1_exit()` を呼び出すと、次のファイルを処理するためには必ず `slc1_init_multi()` を呼び出す必要があります。

9 エラーリファレンス

本製品が返すエラーコードについて説明します。

定義	値 ⁽¹⁰⁾	説明
NORMAL	0	正常終了
BADCODE	1	データが壊れています
ERR_NOMEM	2	領域の確保に失敗しました
NOBUSY	0	圧縮 / 復元処理実行可能状態
BUSY	1	圧縮 / 復元処理実行不可能状態
ERR_INFILE_OPEN	21	入力ファイルオープンエラー
ERR_OUTFILE_OPEN	22	出力ファイルオープンエラー
ERR_FILE_READ	23	ファイルリードエラー
ERR_FILE_WRITE	24	ファイルライトエラー
ERR_INFILE_CLOSE	25	入力ファイルクローズエラー
ERR_OUTFILE_CLOSE	26	出力ファイルクローズエラー

10 PC版評価プログラムについて

本製品に添付している実行形式ファイル(Slc_IS.exe)は、本製品の動作を確認するための評価用アプリケーションです。

・起動方法

```
prompt> Slc_IS option infile outfile [ rate [buff_size]]
```

option **書式** {E[{C|S|L|D}]{B|L}}|D}¹

圧縮 圧縮処理の場合は、'E'を指定します。
圧縮ファイルの先頭に圧縮ブロック数を付加する場合はオプション1を'E'に続けて記述します。

オプション1	説明
C	char 幅の圧縮ブロック数をファイルの先頭へ付加します。
S	short 幅の圧縮ブロック数をファイルの先頭へ付加します。
L	long 幅の圧縮ブロック数をファイルの先頭へ付加します。
D	long long 幅の圧縮ブロック数をファイルの先頭へ付加します。

オプション1を省略して'E'のみで実行する場合は、圧縮データのみがファイルに出力されます。

圧縮ブロック数を付加する場合に、その数値の出力形式を指定する場合はオプション1に続けてオプション2を記述します。
オプション2を省略した場合は、'B'として処理を行います。

オプション2	説明
B	圧縮ブロック数をビッグエンディアンで出力します。
L	圧縮ブロック数をリトルエンディアンで出力します。

復元 復元処理の場合は、'D'を指定します。

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。

rate 圧縮レベルを0~8で指定します。
数値が大きいほど圧縮率は良くなります。
圧縮と復元には同じ圧縮レベルを指定してください。
省略時は7になります。

¹ 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

buff_size 入出力バッファのサイズを 1024 ~ 32768 で指定します。
数値が大きいほど圧縮率は良くなります。
圧縮と復元には同じサイズを指定してください。
省略時は 32768 になります。

・圧縮処理の例

data.raw を圧縮して data.cmp へ出力します。圧縮レベルは 7 です。

```
prompt> Slc_IS e data.raw data.cmp
```

data.raw を圧縮レベル 5 で圧縮して data.cmp へ出力します。その際に char 幅の圧縮ブロックサイズを付加する場合は、次のように指定します。

```
prompt> Slc_IS ec data.raw data.cmp 5
```

data.raw を圧縮レベル 2 で圧縮して data.cmp へ出力します。その際にリトルエンディアンの short 幅の圧縮ブロックサイズを付加する場合は、次のように指定します。

```
prompt> Slc_IS esl data.raw data.cmp 2
```

data.raw を圧縮して data.cmp へ出力します。圧縮レベルは 7 で、バッファサイズは 16384 です。

```
prompt> Slc_IS e data.raw data.cmp 7 16384
```

・復元処理の例

圧縮レベル 5 で圧縮された data.cmp を復元して data.bin へ出力する場合は、次のように指定します。

```
prompt> Slc_IS d data.cmp data.bin 5
```

圧縮レベル 7 でバッファサイズが 16384 で圧縮された data.cmp を復元して data.bin へ出力する場合は、次のように指定します。

```
prompt> Slc_IS d data.cmp data.bin 7 16384
```

FUJITSU