

データ圧縮ライブラリ

# ESLC for WindowsCE

ユーザーズマニュアル

03 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

製品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、WindowsCE、Visual C++は Microsoft Corporation の商標または登録商標です。

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。

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 .....	2
4.4	入出力バッファ .....	2
4.5	ライブラリが使用する外部関数について.....	2
5	圧縮データの構造 .....	3
6	定義.....	4
6.1	定数 .....	4
6.2	関数 .....	4
7	関数.....	5
7.1	slc1_init().....	6
7.2	slc1_encode().....	7
7.3	slc1_decode().....	8
8	使用例 .....	9
8.1	圧縮処理 .....	9
8.2	復元処理 .....	10
8.3	添付のサンプルソース.....	11
9	エラーリファレンス .....	12
10	PC 版評価プログラムについて.....	13

## 1 概要

本製品は、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ファイルが同封されています。

- eslc\_ed7.lib                      圧縮・復元両用ライブラリ（圧縮レベル7）
- eslc\_d7.lib                        復元専用ライブラリ（圧縮レベル7）
- eslc.h                              C言語用ヘッダファイル

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

本製品の動作確認環境は、次のとおりです。

OS     : **WindowsCE 5.0** 日本語版

## 3 開発環境

本製品は、Microsoft 社製の開発ツールであるeMbedded Visual C++ 4.0 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、eMbedded Visual C++ 4.0 を使用してください。

## 4 組み込み方法

ライブラリを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.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“\_ANSI”をシンボル定義してください。

### 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。

### 4.4 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。

### 4.5 ライブラリが使用する外部関数について

ライブラリでは、次の標準関数を使用しています。

- ・ char \*memcpy( char \*s1, char \*s2, int c, int n );
- ・ char \*memset( char \*s, int c, int n );

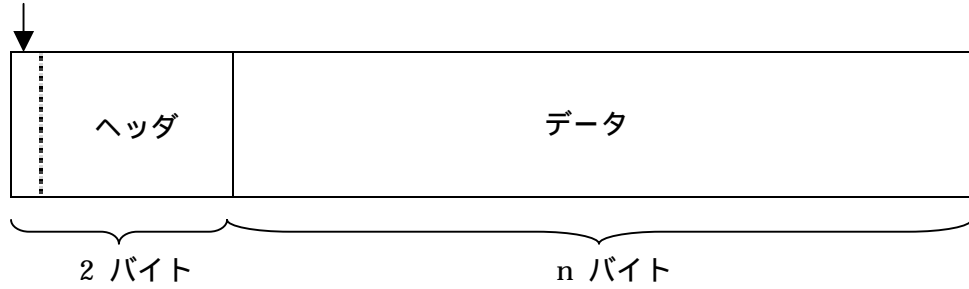
リンク時にこれらでシンボル参照エラーが発生した場合は、ユーザ側で同等の関数を準備してください。

## 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.hで定義しています。

### 6.1 定数

NORMAL	0
BADCODE	1
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 関数

関数名	説明
slc1_init()	内部データ領域の初期化を行います。
slc1_encode()	データを圧縮します。
slc1_decode()	圧縮データを復元します。

## 7 関数

本製品では、次の3つの関数を提供しています。

slc1\_init()

slc1\_encode()

slc1\_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()

[機能]

内部データ領域の初期化を行います。

[形式]

```
int SLCDECLARE slc1_init(  
unsigned short      buff_size  
)
```

[引数]

buff\_size 1回の処理サイズ (1024 ~ 32768)

[戻り値]

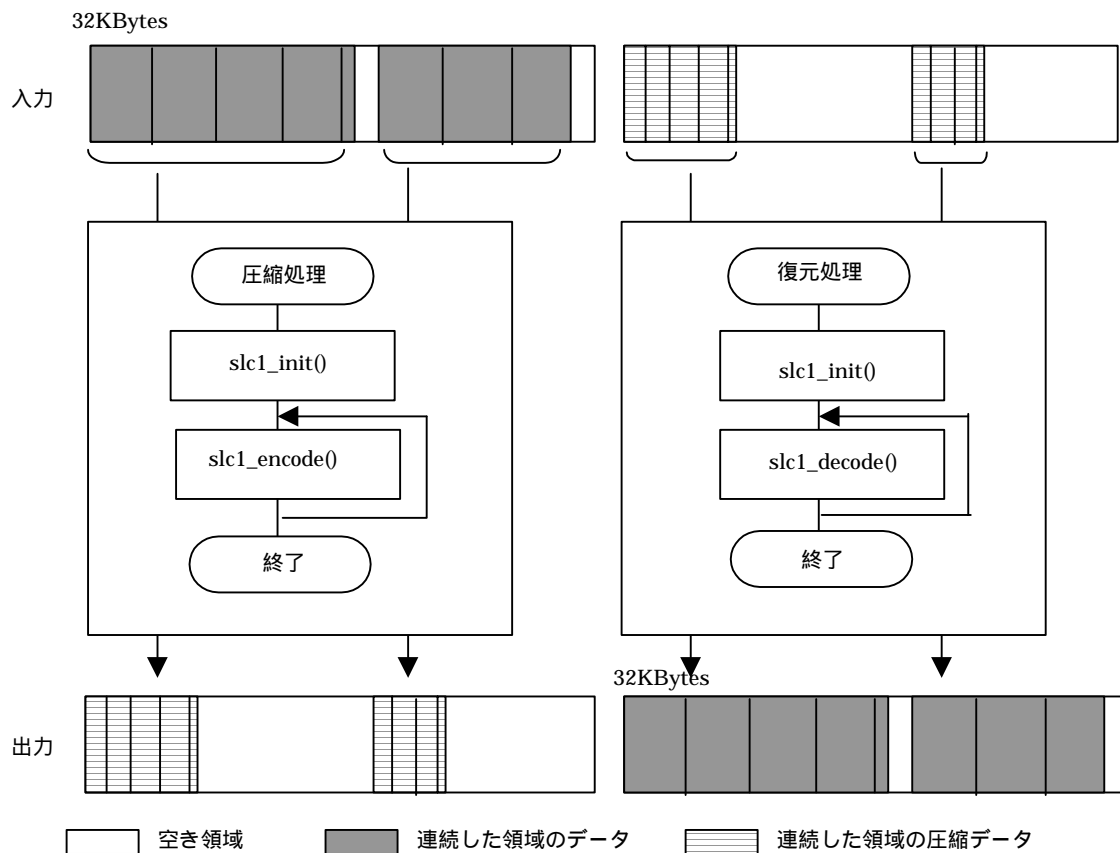
int == NORMAL 正常終了  
 != NORMAL エラー

[説明]

内部データ領域の初期化を行います。

連続した領域のデータを処理するとき、最初に使用するslc1\_encode()、slc1\_decode()の前に呼び出してください。つまり、連続した領域の処理毎に初期化が必要です。

**buff\_size**には、1回の圧縮処理 (slc1\_encode()呼び出し) で処理するデータ量を指定します。復元処理 (slc1\_decode()呼び出し) でも同じ値を指定する必要があります。



## 7.2 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  
)
```

### [引数]

InBuf	オリジナルデータを格納した領域へのポインタ
OutBuf	圧縮データを格納する領域へのポインタ
inlen	オリジナルデータのバイト数 (<= 32768)
outlen	[in] 無視 [out] 圧縮データのバイト数

### [戻り値]

int	== NORMAL	正常終了
	== BADCODE	エラー (inlen = 0)

### [説明]

**InBuf** から **inlen** バイト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。

### 7.3 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  
)
```

[引数]

InBuf	圧縮データを格納した領域へのポインタ
OutBuf	オリジナルデータを格納する領域へのポインタ
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] 復元したオリジナルデータのバイト数

[戻り値]

int	==	NORMAL	正常終了
	!=	NORMAL	エラー

[解説]

**InBuf** から圧縮データをから読み取り、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。

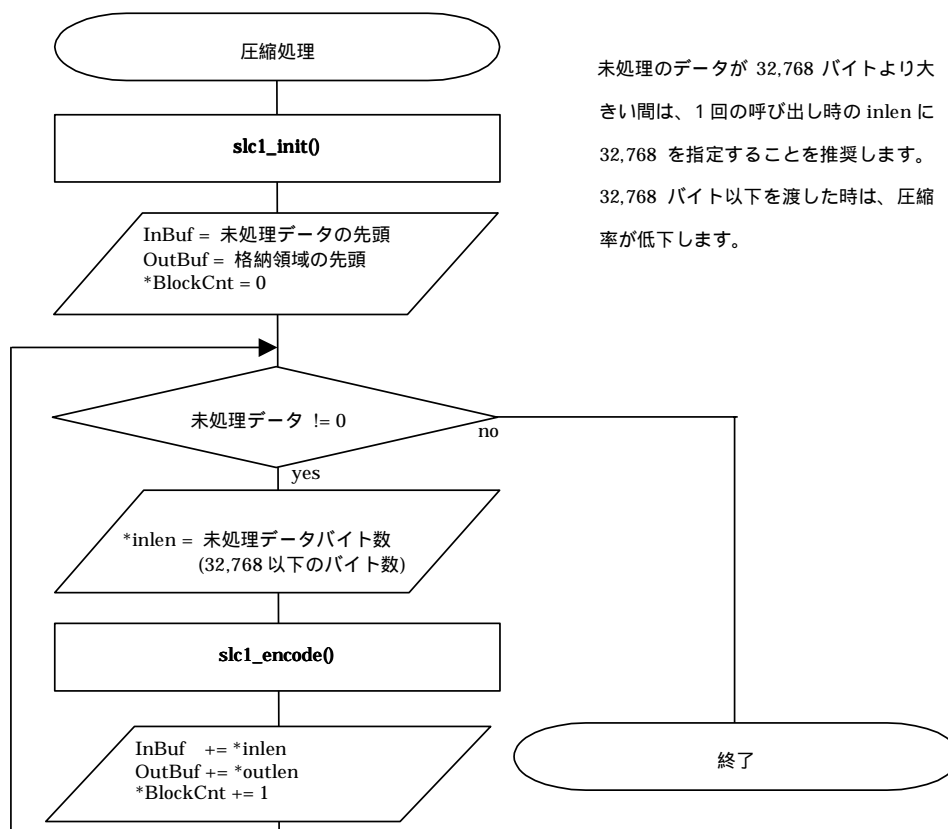
## 8 使用例

本製品のインタフェースを使用して圧縮処理と復元処理の例を示します。

復元処理時には、圧縮データのブロック数 (slc1\_encode() の呼び出し回数) が必要なため、圧縮処理時に圧縮データのブロック数を記録します。復元処理時は、このブロック数を使用して全圧縮データを復元するのに必要なslc1\_decode() の呼び出し回数を知ることができるようにしています。

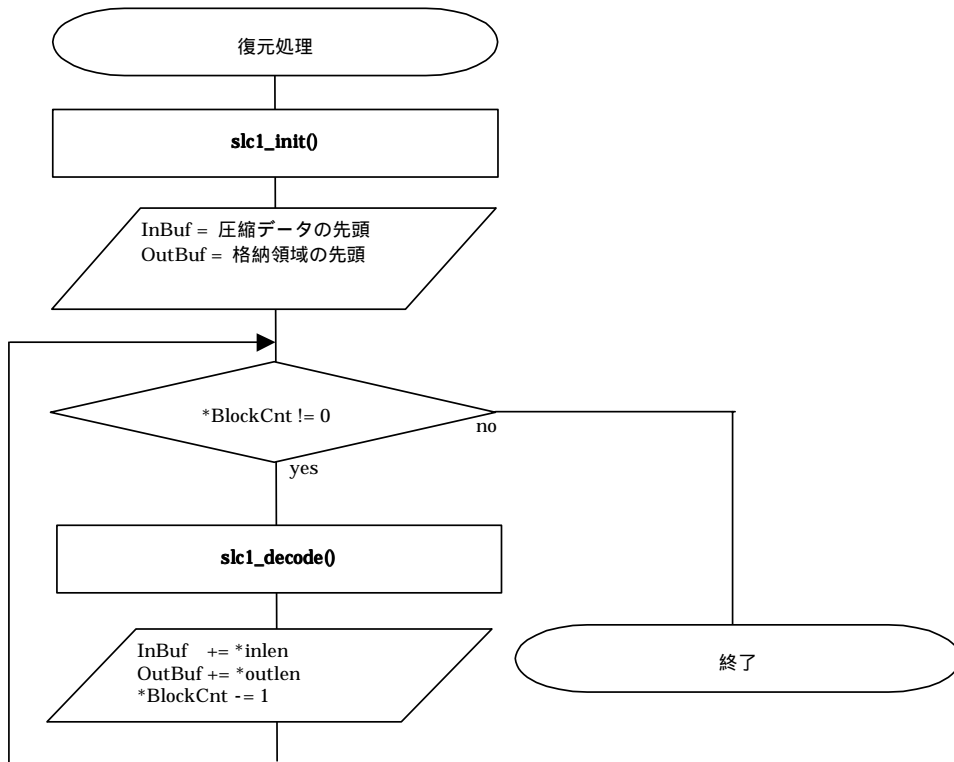
### 8.1 圧縮処理

1. 処理の開始時にslc1\_init()を呼び出します。  
この時、1回の処理サイズを設定します。
2. 次の各ポインタ変数に初期値を代入します。
  - ・ InBuf には、圧縮するデータ領域の開始アドレス
  - ・ OutBuf には、圧縮されたデータの格納領域の開始アドレス
  - ・ BlockCnt には、圧縮ブロック数格納領域のアドレス
3. その後、未処理のデータが無くなるまでslc1\_encode()の呼び出しと、InBuf と OutBuf の更新、\*BlockCnt のインクリメントを繰り返します。



## 8.2 復元処理

1. 処理の開始時にslc1\_init()を呼び出します。  
この時、1回の処理サイズを設定します。
2. 次の各ポインタ変数に初期値を代入します。
  - ・ InBuf には、圧縮データ領域の開始アドレス
  - ・ OutBuf には、復元したデータを格納する領域の開始アドレス
  - ・ BlockCnt には、圧縮ブロック数格納領域のアドレス
3. その後、\*BlockCnt 値が 0 になるまでslc1\_decode( )の呼び出しと、InBuf と OutBuf の更新、\*BlockCnt のデクリメントを繰り返します。



### 8.3 添付のサンプルソース

PC 版評価プログラム(Slc\_IS.exe)で圧縮データのブロック数を出力させた場合は、圧縮データのブロック数が圧縮データの前に配置されます。本製品に添付されているサンプルソース(`smp1_enc.c`, `smp1_dec.c`)は、これと同じ形式のデータを扱えるようになっていますので、参考にしてください。

## 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 <sup>(10)</sup>	説明
NORMAL	0	正常終了
BADCODE	1	データが壊れています

## 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。復元のためのライブラリを使用する場合は、圧縮データを作成するためにも使用できます。

### ・ 起動方法

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

option 書式 {E [{C | S | L | D}]{B | L}} | D<sup>1</sup>

**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	
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。

<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

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