

ESLC
for
SH

ユーザーズ マニュアル

第 04 版

富士通デバイス株式会社

ESLC for SH の著作権は富士通デバイス株式会社が保有しています。

SuperH は株式会社ルネサス テクノロジーの商標です。

SuperH RISC engine family C/C++ コンパイラ パッケージは株式会社ルネサス テクノロジーの製品です。

1. 本書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、本誌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. 本書に記載した内容を、弊社に無断で転載または複製することをご遠慮ください。
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。

目次

1	ライブラリの構成.....	1
2	ライブラリの作成環境.....	2
3	ライブラリ組み込みの注意事項.....	2
3.1	ヘッダファイルのインクルードについて.....	2
3.2	コンパイル時のデファイン指定について.....	2
3.3	データ圧縮ライブラリのリンクについて.....	2
3.4	ライブラリが使用する外部関数について.....	2
3.5	slc1_init()の引数 buff_size に与える 1 回の処理サイズについて.....	3
4	定義.....	4
4.1	定数.....	4
4.2	関数.....	4
5	関数.....	5
5.1	slc1_init().....	6
5.2	slc1_encode().....	7
5.3	slc1_decode().....	8
6	使用例.....	9
6.1	圧縮処理.....	9
6.2	復元処理.....	10
6.3	添付のサンプルソース.....	10
7	エラーリファレンス.....	11
8	PC・Solaris/SPARC 版評価プログラムについて.....	12

1 ライブラリの構成

次のSH3/SH3E/SH4用ライブラリと PC 版評価プログラムが同封されています。

・ SLC_IS.EXE	PC 版評価プログラム
・ slc_is	Solaris/SPARC 版評価プログラム
・ C-V6¥	C コンパイラ V6 用のライブラリを格納
・ SH3¥	SH3/SH3DSP 用のライブラリを格納
・ SH3E¥	SH3E 用のライブラリを格納
・ SH4¥	SH4 用のライブラリを格納
・ ES LC_ED.lib	圧縮・復元両用ライブラリ (圧縮レベル 7)
・ ES LC_ED6.lib	圧縮・復元両用ライブラリ (圧縮レベル 6)
・ ES LC_ED2.lib	圧縮・復元両用ライブラリ (圧縮レベル 2)
・ ES LC_ED0.lib	圧縮・復元両用ライブラリ (圧縮レベル 0)
・ ES LC_D.lib	復元専用ライブラリ (圧縮レベル 7)
・ ES LC_D6.lib	復元専用ライブラリ (圧縮レベル 6)
・ ES LC_D2.lib	復元専用ライブラリ (圧縮レベル 2)
・ ES LC_D0.lib	復元専用ライブラリ (圧縮レベル 0)
・ C-V7¥	C コンパイラ V7 用のライブラリを格納
・ SH3¥	SH3/SH3DSP 用のライブラリを格納
・ SH3E¥	SH3E 用のライブラリを格納
・ SH4¥	SH4 用のライブラリを格納
・ ES LC_ED.lib	圧縮・復元両用ライブラリ (圧縮レベル 7)
・ ES LC_ED6.lib	圧縮・復元両用ライブラリ (圧縮レベル 6)
・ ES LC_ED2.lib	圧縮・復元両用ライブラリ (圧縮レベル 2)
・ ES LC_ED0.lib	圧縮・復元両用ライブラリ (圧縮レベル 0)
・ ES LC_D.lib	復元専用ライブラリ (圧縮レベル 7)
・ ES LC_D6.lib	復元専用ライブラリ (圧縮レベル 6)
・ ES LC_D2.lib	復元専用ライブラリ (圧縮レベル 2)
・ ES LC_D0.lib	復元専用ライブラリ (圧縮レベル 0)
・ eslc.h	C 言語用ヘッダファイル
・ smpl_enc.c	圧縮用サンプルソース
・ smpl_dec.c	復元用サンプルソース
・ smpl_cmd.c	ファイル操作イメージのサンプルソース

圧縮・復元両用ライブラリは、圧縮・復元の両方の機能を備えています。また、復元専用ライブラリは復元機能のみを備えています。

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。圧縮と復元は同じレベルのものを使用する必要があります。必要となる RAM 量の目安は以下のとおりです。

レベル 7	圧縮・復元両用	約 273Kbytes	復元専用	約 144Kbytes
レベル 6	圧縮・復元両用	約 145Kbytes	復元専用	約 81Kbytes
レベル 2	圧縮・復元両用	約 24.6Kbytes	復元専用	約 20.5Kbytes
レベル 0	圧縮・復元両用	約 18.6Kbytes	復元専用	約 17.5Kbytes

2 ライブラリの作成環境

株式会社ルネサス テクノロジーの開発ツールであるSuperH RISC engine C/C++ family コンパイラを用いてコンパイルしてあります。ユーザプログラム作成には、SuperH RISC engine C/C++ family コンパイラを利用してください。

(補足 1) Ver6 用は、下記製品に搭載されているコンパイラを使用しています。

SuperH RISC engine family C/C++ コンパイラ パッケージ V6.0AR2

(補足 2) Ver7 用は、下記製品に搭載されている HEW 環境内のコンパイラを使用しています。

SuperH RISC engine family C/C++ コンパイラ パッケージ V7.1.00

3 ライブラリ組み込みの注意事項

ライブラリ組み込みの注意事項を説明します。

3.1 ヘッドファイルのインクルードについて

ソースファイルへeslc.hをインクルードします。

3.2 コンパイル時のデファイン指定について

コンパイル時のデファイン指定でSHと_ANSIを指定します。

3.3 データ圧縮ライブラリのリンクについて

- ・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。
- ・本ライブラリでは、次のセクションを使用しています。適切な領域への割り付けを行ってください。

Peslc	Program セクション(ROM)
Ceslc	Const セクション(ROM)
Beslc	初期値なし Data セクション(RAM) できるだけ高速な RAM に割り当ててください。

3.4 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 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。

3.5 slc1_init()の引数 buff_size に与える 1 回の処理サイズについて

本ライブラリは、一定処理サイズ毎に処理を行います。この処理サイズは、slc1_init()の引数 buff_size で指定します。

1 回の圧縮処理では、この指定されたサイズを圧縮元の最大データ量として圧縮を行います。1 回の復元処理でも同様に、指定されたサイズを復元最大データ量として復元を行います。圧縮時と復元時には、必ず同じ値を指定してください。この値は、小さくするほど圧縮率は悪くなりますので、通常は最大値 (SLC1_BUF_SIZE : 32768 バイト) を利用することをお勧めします。

4 定義

ESLC for SHの定義について説明します。ここで説明している定義を使用するには、`eslc.h`をインクルードしてください。

4.1 定数

NORMAL	0
BADCODE	1
SLC1_BUF_SIZE	(unsigned int) ((unsigned int) 8192*4)

4.2 関数

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

5 関数

関数は、内部データの初期化、圧縮、復元の関数があります。
提供する関数名は、次のとおりです。

`slc1_init()`

`slc1_encode()`

`slc1_decode()`

5.1 slc1_init()

[形式]

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

[引数]

buff_size 1回の処理サイズ (1024 ~ 32768)

[戻り値]

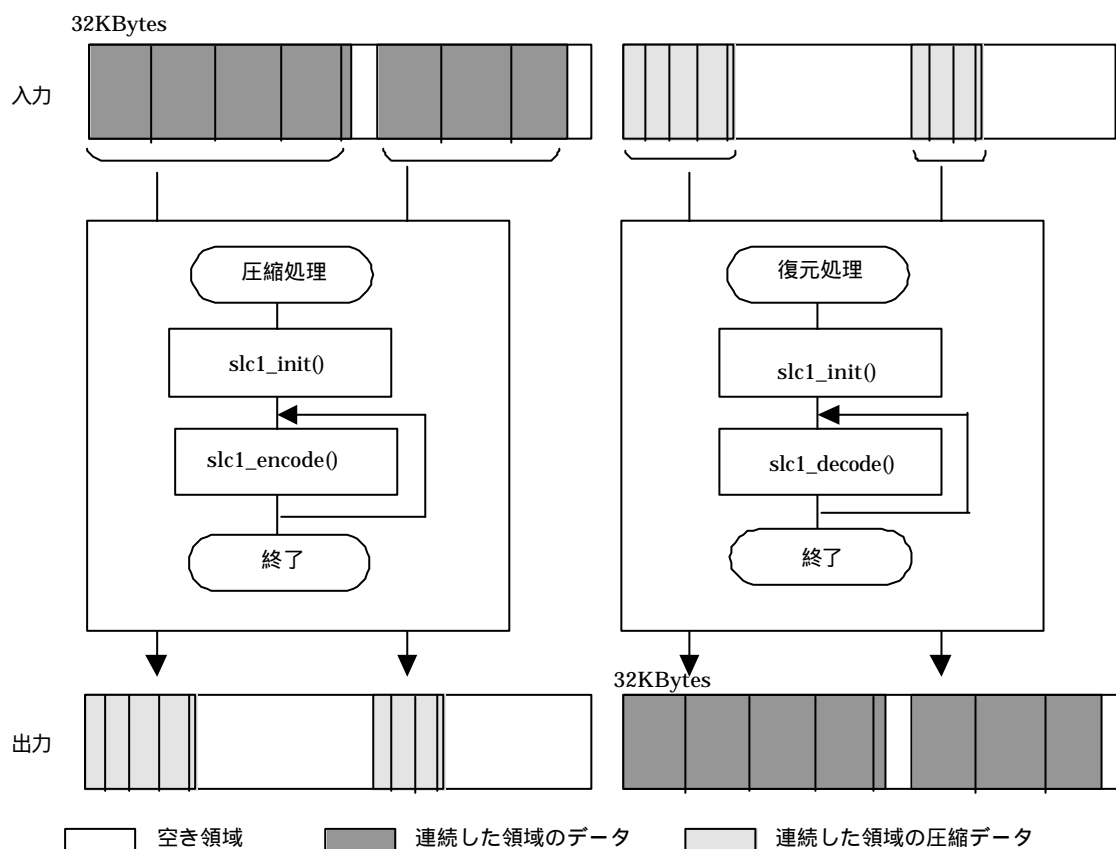
int == NORMAL 正常終了
 != NORMAL エラー

[解説]

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。



5.2 slc1_encode()

[形式]

```
int slc1_encode(  
unsigned char *      InBuf  
unsigned char *      OutBuf  
unsigned short       inlen  
unsigned short *     outlen  
)
```

[引数]

InBuf	圧縮するデータ領域へのポインタ
OutBuf	圧縮したデータを格納する領域へのポインタ
inlen	圧縮するデータのサイズ(< 32768)
outlen	圧縮したデータのサイズ

[戻り値]

int	==	NORMAL	正常終了
	!=	NORMAL	エラー

[解説]

データを圧縮します。

InBuf から **inlen** バイト読み取り、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。

5.3 slc1_decode()

[形式]

```
int slc1_decode(  
unsigned char *      InBuf  
unsigned char *      OutBuf  
unsigned short *     inlen  
unsigned short *     outlen  
)
```

[引数]

InBuf	圧縮データ領域へのポインタ
OutBuf	復元データを格納する領域へのポインタ
inlen	復元するデータのサイズ
outlen	復元したデータのサイズ

[戻り値]

int	==	NORMAL	正常終了
	!=	NORMAL	エラー

[解説]

圧縮データを復元します。

圧縮データを **InBuf** から読み取り、復元したデータを **OutBuf** へ書き込みます。

読み取った圧縮データのバイト数は、**inlen** へ書き込みます。

復元データのバイト数は、**outlen** へ書き込みます。

6 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、圧縮時に圧縮データのブロック数を記録します。復元処理の時は、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。

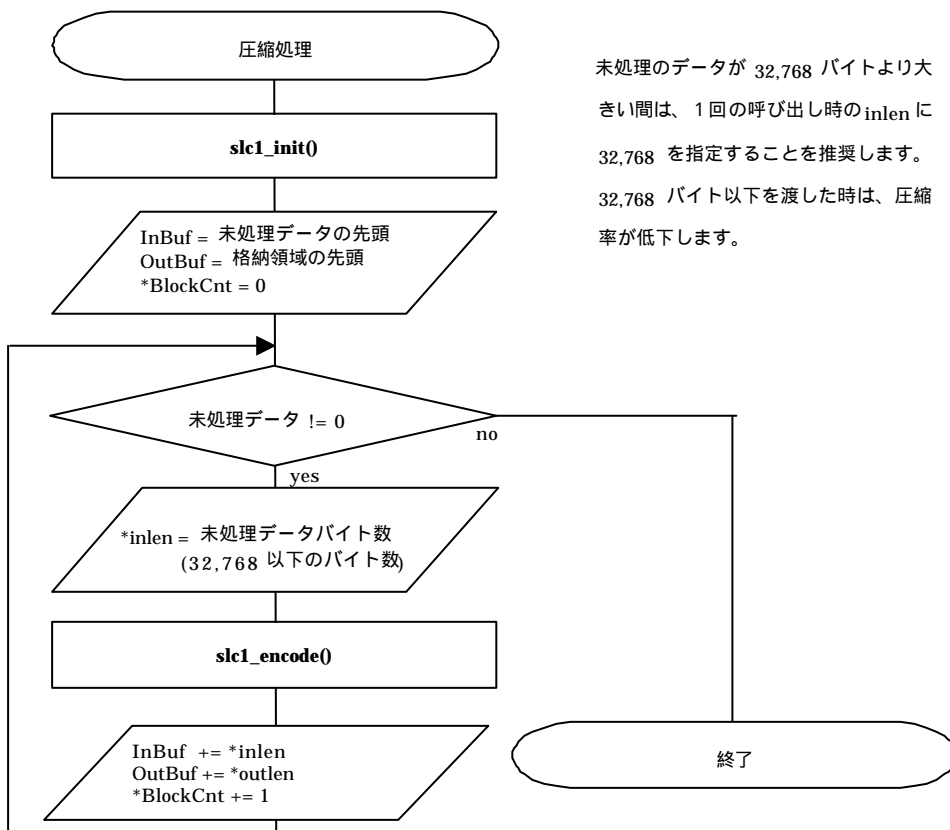
6.1 圧縮処理

処理の開始時に `slc1_init()` を呼び出します。

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

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。



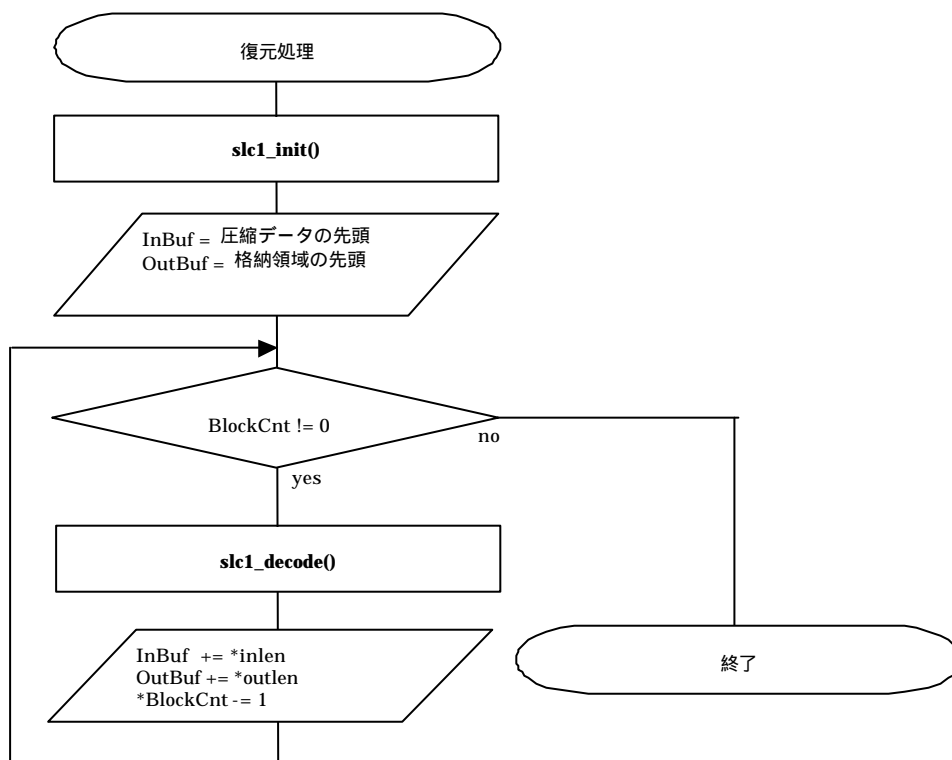
6.2 復元処理

処理の開始時にslc1_init()を呼び出します。

次の各ポインタ変数に初期値を代入します。

- ・ BlockCnt には、圧縮ブロック数格納領域のアドレス
- ・ 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

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。



6.3 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 (smpl_enc.c, smpl_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。

7 エラーリファレンス

ESLC for SHが返すエラーコードについて説明します。

これらの定義は、eslc.hで定義しています。

定義	値 ⁽¹⁰⁾	説明
NORMAL	0	正常終了
BADCODE	1	データが壊れています

8 PC・Solaris/SPARC 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 / slc_is)は、ESLC for SH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]**

option 書式 {E[{C|S|L|D}]{B|L}}|D¹

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ク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 圧縮レベル番号を 0~8 で指定します。数値が大きいほど圧縮率は上がりますが、必要 RAM サイズは大きくなります。選択した圧縮ライブラリの圧縮レベル番号と同じ物を指定してください。省略時は 7 になります。
rate は、圧縮'E'および復元'D'指定時に有効です。

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

・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
```

・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
```


FUJITSU