2014/12

# EL ドングルマニュアル プログラム開発編

有限会社リビッグ 横浜市港南区上大岡西 1-12-2-801 Tel: 045-843-7122 Fax: 045-843-7142 <u>http://www.ribig.co.jp/el/</u>

## 目次

EL について	3
コードポート技術	3
開発手順例	3
スマートカード	4
自動セルフロッキング機構	4
GUSN(グローバルー意シリアル番号)	4
暗号処理コプロセッサ	4
内臓タイマー	4
HID モード/専用ドライバモード	4
EL 向けプログラム開発	5
EL 内部で実行するプログラム	5
コンピュータ側から EL デバイスを操作するプログラム	5
開発に必要な EL に関する知識	7
ファイルシステム	7
ファイルタイプ	7
1.実行ファイル	8
2. データファイル	8
3.鍵ファイル	8
権限と読み書き属性	8
PIN 認証[コンピュータ側からの操作が対象となります]	9
1. 開発者 PIN	9
2. ユーザ PIN	9
3. PIN の設定先	9
4. PIN ロック	10
5. PIN 初期化	10
EL プログラム作成の概要	.11
1. 通常の C プログラムを作成	.11
2. EL にコードポートする部分を選択。	.11
3. EL プログラムのビルド	13
4.EL ヘコードポート	13
5.コンピュータ側のプログラム変更	15
EL 用プログラム開発の詳細	18
プログラムの基本構造	18
実行モード	18

EL のメモリ構造	
メモリの割り当て	
メモリアドレス	19
メモリ共有	
コードメモリ	
アライメントの問題	
ビックエンディアン と リトルエンディアン	
ライブラリファイル	
EL プログラム作成時の留意点	
コンピュータ側からの EL へのアクセス	
1. 一般アクセス(無認証アクセス)	
<b>2.</b> 開発者レベルアクセス	
<b>3.</b> ユーザレベルアクセス	31
Keil C51 の使い方	33
Keil C51 の特徴	33
EL 開発用パッチ	33
Keil C51 の IDE - uVision	
プロジェクト作成	
プログラムのビルトとデバッグ	
付属ツール	45
HEX ファイル変換ツール	45
EL デバイス設定ツール (DevTest.exe )	45
1. EL デバイスの初期化 / 再初期化	47
2. ファイルのダウンロード	
3. 実行ファイルの起動	50
4. ディレクトリのクリア	51
5. <b>PIN</b> の変更	52
6.RSA 鍵の生成	53

ELについて

Elite EL シリーズは強力で効果的なプロテクションを実現するドングルです。コードポートとスマートカードの組み合わせによってハイエンド市場向けのプロテクトドングルに必要な機能を提供します。

コードポート技術



コード(プログラム)を ELにポート(転送)すると、EL内では独立した実行ファイル (EXF) として保存されます。コンピュータ側プログラムは、引数を付けて EL 内の実行ファイル (EXF)を呼び出すことができます。呼び出された EXF は EL 内の OS によって実行され、 処理終了後に呼び出し元に戻り値を返します。EL が接続されていなければ、コンピュータ 側のプログラムは実行を続けるために必要なコードが実行されないため、正常に動作しま せん。

EL には複数プログラム (コード) をポートできます (EL 内には複数の EXF を保存できま す)。 EXF は C で作成します。PC 用の標準関数を使った C プログラムは、わずかな変更 で EXF 作成に再利用できます。

<u>開発手順例</u>

- 1. プロテクトを考えずに C/C++でコンピュータ側のアプリケーションプログラム を作成します。
- 作成したコードの一部を EL で実行させることを考えます。EL で実行させるコード部分を取り出して、EL プログラムを作成します。最終的には EL の CPU 用の C クロスコンパイラでコンパイルして、EL にコードポートします。

3. コンピュータ側のアプリケーションプログラムでは、EL に移した部分を EL 内プ ログラムの呼び出しに置き換えます。

## スマートカード

NXP(Philips) の 16bit CPU,RAM,EEPROM, USB 通信モジュールをすべて搭載したスマ ートカードを心臓部に使っています。スマートカード(IC カード)のため、各種ハードウ ェア攻撃に対して有効的な耐性があります。利用しているスマートカードは EAL 5 以上の 評価保障レベルに準拠しています。

## 自動セルフロッキング機構

スマートカードへのアクセスは PIN によって保護されています。一定回数以上の誤った PIN でアクセスを試みると、スマートカードは自動ロックして外部からアクセスできない ようになります( 解除はできません )

#### GUSN(グローバルー意シリアル番号)

スマートカードには変更不可、上書き不可の GUSN が設定されています。

#### 暗号処理コプロセッサ

EL は RSA(1024bit)/TDES 暗号処理用のコプロセッサを搭載しているため、暗号処理は 高速です。また、ハードウェアによる真の乱数を生成できます。

## 内臓タイマー

ELにはタイマーを内蔵した機種があります。

#### HID モード/専用ドライバモード

ELは HID モードかドライバモードで動作します。HID モードでは、EL は OS のドライバ によって自動認識されます。ドライバモードの EL を利用するには専用ドライバをインスト ールする必要があります。ドライバモードでは、複数のプロセスが同時に EL を呼び出せ ますが、HID モードでは複数のプロセスが EL にアクセスすることはできません。

## EL 内部で実行するプログラム

EL で実行するプログラムは Keil C51 を使って IDE(統合開発環境)環境で開発できます。 Kei C51 に関する詳細は Keil Software の Web ページ <u>http://www.keil.com</u>をご覧く ださい。

## <u>コンピュータ側から EL デバイスを操作するプログラム</u>

付属 C 用の API を使って各種言語で作成したプログラムから EL デバイスを操作します。



ELの機能を利用するには、EL内で実行するプログラムを作成します。

- ◆ EL内で実行するプログラムは、ELのOS提供関数やC標準関数を呼び出すことができます。
- ◆ 各種ファイルを作成できます。
- ◆ SDK に含まれる Keil C51 向けパッチをあてると IDE 環境で EL プログラムをデバッ グできるようになります。

コンピュータ側のプログラムは、EL内のプログラムを呼び出すようにします。

- コンピュータ側では EL を操作するための API や暗号補助関数を使って EL 機能を利 用したプログラムを作成します。
- 付属の DevTool( DevTest.exe )プログラムをつかって、GUI で EL のファイルシス テムに対する操作を行えます。

このマニュアルでは、EL内部で実行するプログラムを作成するために必要な EL に関する 知識、プログラム作成/デバッグ方法、EL デバイスへのプログラムのポート、そして、コン ピュータ側から EL 内部プログラムを呼び出す方法について説明します。コンピュータ側 からの EL 操作には DevTest.exe を使います。

このマニュアルとは別に、SDK には豊富な EL プログラムのサンプルがケーススタディと いう形式で用意されています。

コンピュータ側プログラムによる EL操作は、主に EL内プログラムの呼び出しになります。 開発者側での EL設定には、API による EL操作が必要になるはずです。ELに対する操作 APIは S4XXXX 関数のマニュアルをご参照ください。

## 開発に必要な EL に関する知識

ファイルシステム

EL はディレクトリ - ファイルの階層構造で管理します。各ディレクトリには ディレクト リ ID を割り当てます。ディレクトリに重複した ディレクトリ ID を割り当てることはで きません。

1つのルートディレクトリと 3 階層までのディレクトリを作成できます。ファイルシステムがデイバスのすべてのストレージメモリを占有します。管理が複雑になるため、本当に必要なケースを除き、複数のディレクトリは作らずに単一ディレクトリで使うことをお勧めします。

ディレクトリ作成時にディレクトリが占めるメモリサイズを指定します。ディレクトリ作 成後、ディレクトリサイズは変更できません。

ルートディレクトリを削除すると、ディレクトリ階層のファイルとディレクトリすべてが 削除されます。

サブディレクトリは直接削除できません。内容がクリアされるだけです。ディレクトリがク リアされると、ディレクトリ内のファイルやディレクトリは削除されます。また指定ディレ クトリの開発者 PIN とユーザ PIN はリセットされます。ディレクトリを削除して、占めて いる領域を解放するには、上のディレクトリをクリアします。

ファイルタイプ

ELには3つのファイルタイプがあります。

- 1. 実行ファイル(EXF)
- 2. データファイル
- 3. 鍵ファイル

すべてのファイルは外部から直接、<u>読み出す</u>ことはできません。開発者 PIN を指定することで、ファイル作成 / 変更は直接外部から行えます。

1. 実行ファイル

実行ファイルは内臓 OS によってロード、実行されます。同じディレクトリにある他のファ イルに対して OS 提供のシステム関数を使って操作(作成、読み書き)できます。ただし、 他の実行ファイルに対する操作は、その実行ファイルの読み書き属性に影響を受けます。

2. データファイル

データファイルはバイナリ形式のみサポートします。外部からデータファイルを直接読み 込むことはできません。データを読み出すには、EL内のプログラム経由で行います。

3. 鍵ファイル

RSA の鍵ファイルです。詳細は SES RSA 関連関数をご参照ください。

#### 権限と読み書き属性

ファイル	タイプ	開発者 PIN	ユーザ PIN	EXF
実行ファイル	読み書き禁止	W	E	Ν
	読み書き許可	W	E	R/W
データファイル		W	Ν	R/W
鍵ファイル		W	Ν	W

R: 読み込み W:書込み E:実行 N:不可

ファイル操作は、コンピュータ側からの操作なのか、EL内部での操作なのか明確にするようにしてください。この区別を意識しないと混乱の元となります。

[コンピュータ側からの操作]開発者 PIN で EL にログインすると、すべてのファイルに対して書き込み操作が可能

[コンピュータ側からの操作]ユーザ PIN でログインすると、実行ファイルを実行できますが、データファイルと鍵ファイルに対する操作は行えません。

[EL 内部の操作]実行ファイルは、データファイルの読み書き、鍵ファイルの書込み、読み 書きが許可された実行ファイルに対する読み書きの操作が行えます。

#### PN 認証[コンピュータ側からの操作が対象となります]

2 種類の PIN があります。<u>外部から EL を操作するには</u> PIN でログインしなければな りません。ただし、コンピュータに接続している EL の検索、オープン、ディレクトリ変 更、クローズ、コントールコード送信などには PIN ログインは不要です。

- 1. 開発者 PIN
- 2. ユーザ PIN

#### 1. 開発者 PIN

開発者 PIN は 24 バイトです。既定値は "123456781234567812345678" です。開発 者 PIN でログインすると、ファイルの読み出しはできませんが、EL に対するその他のす べての操作が行えます。このため開発者以外が開発者 PIN を入手できないようにしてく ださい。

- 開発者 PIN は、開発側で EL を設定するときだけ必要です。絶対に開発者 PIN を配布プログラム内に記述するようなことはしないでください。
- EL のすべてのディレクトリの開発者 PIN は、ユーザに渡す前に必ず既定値 から別の値に変更してください。

外部に開発者 PIN が知られてしまっても、簡単に EL 内のデータが読みだされることはあ りません。直接読み込むことは不可能です。EL 内のファイル名の一覧を得る手段は用意さ れていません。しかし、プログラムの書き換え、ディレクトリ削除などどのような事態が 発生するか予測できません。開発者 PIN 管理には細心の注意を払ってください。

#### 2. ユーザ **PIN**

ユーザ PIN は 8 バイトです。既定値は "12345678"です。ユーザ PIN で行えるのは、 EL 内の実行ファイル(EXF)を呼び出すことだけです。EL プログラム呼び出し前のユー ザ確認をするための PIN で、絶対に外部に分かってしまってはいけないようなものでは ありません。

#### 3. **PIN**の設定先

PIN は各ディレクトリに独立して設定します。デバイス全体に対して設定されませんので ご注意ください。それぞれのディレクトリは開発者 PIN とユーザ PIN を持ちます。ル ートディレクトリの 開発者 PIN を変更してもサブディレクトリの 開発者 PIN は変更 されません。また、ルートディレクトリに開発者 PIN でログインしても、サブディレク トリに開発者 PIN でログインしたことにはなりません (ELを単一ディレクトリで利用す るように推奨する理由は複雑な PIN 管理を避けるためです)。 複数のプログラムが1つの EL ドングルを利用するときに、それぞれのプログラムが異なるディレクトリを利用すると(異なる開発者 PIN/ユーザ PIN)、プログラム間の干渉を避けることができます。

## 4. PIN ロック

15 回連続して<u>ルートディレクトリ</u>に誤った開発者 PIN でログインしようとすると、EL は ロックします。ロックは解除できません。<u>サブディレクトリ</u>に誤った開発者 PIN でログイ ンするとディレクトリがロックしますが、ディレクトリは削除できます。ユーザ PIN はロ ック動作を引き起こしません。

#### 5. PIN 初期化

ルートディレクトリを削除、再作成したり、サブディレクトリをクリアしたりすると、そのディレクトリの開発者/ユーザ PIN は規定値に戻ります。ファイルシステムに変更した 場合、必ず PIN を既定値から変更することを忘れないでください。

```
1. 通常の C プログラムを作成
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
/*bubble sort function*/
void bubble_sort(unsigned char *p, int len)
{
       int i,j;
       unsigned char tmp;
       for (i=0; i<len-1; i++)
       {
               for (j=0; j<len-i-1; j++)
       {
                       if (p[j] < p[j+1])
                       {
                               tmp = p[j];
                               p[j] = p[j+1];
                               p[j+1] = tmp;
                       }
               }
       }
}
/*main procedure*/
void main()
{
       unsigned char test[] = \{4,3,8,2,9,7,1,5,0,6\};
       int len = sizeof(test);
       int i;
       bubble_sort(test, len);
       printf("result: ¥n");
       for (i=0; i<len; i++)
       {
               printf("%d ",test[i]);
       }
}
```

2. EL にコードポートする部分を選択。 その部分を再利用して EL プログラムを作成。 ここでは void bubble\_sort(unsigned char \*p, int len) をEL側に移すことにしたとします <u>ELプログラム</u>

```
#include "ses_v3.h"
/*bubble sort function*/
void bubble_sort(unsigned char *p, int len)
{
        int i,j;
        unsigned char tmp;
       for (i=0; i<len-1; i++)
        {
               for (j=0; j<len-i-1; j++)
                Ł
                       if (p[j] < p[j+1])
                        {
                               tmp = p[j];
                               p[j] = p[j+1];
                                p[j+1] = tmp;
                        }
               }
       }
}
/*EL main procedure*/
void main()
{
        unsigned char *test = pbInBuff;
       int len = bInLen;
       bubble_sort(test, len);
       _set_response(len,test);
       _exit();
}
```

典型的な EL プログラムでは、main() で pbInBuff, pInLen マクロで引数を受け取り、 処理終了後, \_set\_response で結果(バッファとバッファ長)を返してプログラムを \_exit() 終了します。

pbInBuf はコンピュータ側から送られたデータを保持するコミュニケーションバッファを 指しています。pInLen はバッファの長さが入っています。

\_set\_response と \_exit は内臓 OS 提供関数(SES 関数)です。SES 関数を利用するため

ヘッダファイル**"ses\_v3.h"**をインクルードしています。その他の部分は通常の C プログラ ムと同じです。

3. EL プログラムのビルド

uVision IDE でプロジェクトを作成してファイル demo1.c を追加してから、プログラムのソースを入力します。また、ライブラリファイル Ses51L.lib とヘッダファイル ses\_v3.h をプロジェクトディレクトリにコピーしてから、Ses51L.lib をプロジェクトに追加してください。

**Project**->**Build target** でコンパイルして成功すると demo1.hex ができあがります。 詳細は "コード開発"の章をご参照ください。

4. EL ヘコードポート

ELを接続して、EL 開発キットの /tools/devtest.exe を実行してください。



Menu の Reconnect 選択すると Menu で各種操作が行えるようになります。

	Elite EL development and test tool 🛛 – 🗖 🗙
Menu	Reconnect 🚳
Recreate Root Cownload Execute Empty Directory	Reconnect device successfully         Version       :V2.4.0(V0.2.4.0)         Device ID       :526942694730°         Device SN       :980731000000         Left space size:0x1536bytes
PIN Management PC Keys Reconnect	Disconnect to device Would like to have better service, please access our website: http://www.senselock.com

プログラムを転送する前に、ルートディレクトリを作成します。Menu の Recreate Root を選択します。

<b>M</b>	Elite EL development and test tool 🛛 – 🗖 🗙
Menu	Recreate Root 🧖
Recreate Root	Input the developer PIN of root directory:
Download 👸 Execute	Device ID information:
🔊 Empty Directory	Set the information of module license of network dongle
🙀 PIN Management	Delete the root directory only, does not rebuild the file system Before publishing dongles, the developer PIN and user PIN must be changed to non-default value!
PC Keys	Reset device
Reconnect	

開発者 PIN の規定値が自動設定されますので、任意のデバイス ID( 例えば ribig )だけを 入力して "Reset Device"ボタンを押します。 ルートディレクトリ作成後、転送は Download メニュで行います。Demo1.hex をルート ディレクトリに 0xd001 という名前で転送するとします。

<b>1</b>	Elite EL development and test tool $ \Box$ $\times$
Menu	Download 🥸
🧭 Recreate Root	
<b>Š</b>	Download the file path:
Download	Download the name of subdirectory:
Execute	¥ Download file name:
62	d001
Empty Directory	File size(byte):
PIN Management	
	Download the type of file:
PC Keys	This read-writeble for executable file
	Overwrite the original file
Reconnect	download file generate key pair

Browse ボタンで demo1.hex を選択します。ダウンロード先は ¥ とします。ファイル サイズは自動計算されます。既に同じ名前で EL に転送されたプログラムがあれば、 Overwrite the original file にチェックしてください。

```
5. コンピュータ側のプログラム変更
ELに移した部分を EL プログラム呼び出しに置き換えます。
```

```
#include <stdlib.h>
#include <stdlib.h>
#include <stdlib.h>
#include <conio.h>
#include "sense4.h"
/*bubble sort function removed. Add EL invoking code.*/
void call_sense4(char *, unsigned char *, int);
/*main procedure*/
void main()
{
```

```
int len = sizeof(test);
int i;
call_sense4("d001", test, len);
printf("result: ¥n");
for (i=0; i<len; i++)
{
    printf("%d ",test[i]);
}
}
void call_sense4(char *fid, unsigned char *buff, int len)
{
///
}
```

bubble\_sort(unsigned char \*p, int len) 関数呼び出しの代わりに EL側の 実行ファイ ル0xd001 を呼び出すようにします。

```
void call_sense4(char *fid, unsigned char *buff, int len)
{
       SENSE4_CONTEXT ctx = \{0\};
       SENSE4_CONTEXT *pctx = NULL;
       unsigned long size = 0;
       unsigned long ret = 0;
       S4Enum(pctx, &size);
       if (size == 0)
       {
               printf("EL not found!¥n");
               return;
       }
       pctx = (SENSE4_CONTEXT *)malloc(size);
       if (pctx == NULL)
       {
               printf("Not enough memory!¥n");
               return;
       }
       ret = S4Enum(pctx, &size);
       if (ret != S4_SUCCESS)
       {
               printf("Enumerate EL error!¥n");
               free(pctx);
               return;
       }
       memcpy(&ctx, pctx, sizeof(SENSE4_CONTEXT));
       free(pctx);
       pctx = NULL;
```

```
ret = S4Open(\&ctx);
if (ret != S4_SUCCESS)
{
        printf("Open EL failed!¥n");
        return;
}
ret = S4ChangeDir(&ctx, "¥¥");
if (ret != S4_SUCCESS)
{
        printf("No root directory found!\u00e4n");
       S4Close(&ctx);
        return;
}
ret = S4VerifyPin(&ctx, "12345678", 8, S4_USER_PIN);
if (ret != S4_SUCCESS)
{
        printf("Verify User PIN failed!¥n");
       S4Close(&ctx);
        return;
}
ret = S4Execute(&ctx, fid, buff, len, buff, len, &size);
if (ret != S4_SUCCESS)
{
        printf("Execute EL exe failed!¥n");
       S4Close(&ctx);
       return;
S4Close(&ctx);
return;
```

}

呼び出すコードをポートした EL をオープンして、対象ディレクトリに移動します。そして、ユーザPIN でログイン後、Execute でプログラム 0xd001 を呼び出します。

```
一般的なELプログラム呼び出しまでの手続Enum->Open->ChangeDir->VerifyPin->Execute
```

この関数の引数 char \*fid でプログラムファイル名を変えて呼び出すことで、ルートディレクトリの別名のEXFを呼び出すことができます。

このコードをビルドするにはヘッダファイル sense4.h と EL API ライブラリファイル が 必要です。 C言語で作成します。

## プログラムの基本構造

```
#include "ses_v3.h"
#include "string.h"
unsigned char output[256];
unsigned char input[256];
void main()
{
    int input_len = bInLen;
    int output_len = 0;
    memcpy(input, pbInBuff, input_len);
    /* Operate input data here... */
    /* If we get some output data... */
    __set_response((unsigned char)output_len, output);
    __exit();
}
```

pbInBuff, pInLen マクロでコンピュータからの引数を受け取り、処理後、結果を \_set\_response で返します。

データを受け渡すコミュニケーションバッファは 250 バイトを超えることはできません。 250 バイト以上のデータを受け渡しするには、何回かに分けて呼び出さなければなりません。

#### 実行モード

EL の実行ファイル(EXF)は VM 実行ファイルと呼ばれるものです。コンパイル時にコードのモードを VM モード(Virtual Machine Mode)に指定します。

## EL のメモリ構造

ELのRAMメモリは2Kバイトです。オーバフローしないように使わなければなりません。

メモリの割り当て

VM モードではメモリーには内部 RAM と外部 RAM の2種類あります。内部 RAM は 256 バイト、残りが外部 RAM になります。コード内で xdata, idata キーワードを使って、ど ちらのメモリを使うかを指定できます。内部 RAM はプログラム実行中にスタックで使われ ますので、通常はオーバフローを避けるために外部 RAM を使います。

xdata unsigned long x; // 外部RAM idata unsigned long i; // 内部RAM

xdata, idata キーワードは必ず指定する必要はありません。コンパイル時に Memory Model に Large (EL 用プログラムは Large で作成するようにします)を設定しておく と、すべての変数は外部 RAM に配置されます。内部 RAM に配置する変数にだけ idata を 指定します。

メモリアドレス

VMモードでのメモリアドレスは以下の通りです。

0 0xFF	0 0x7FF	0x8000	0x8001
			0x80FA
内部RAM	外部RAM	データ長	コミュニケーション
			バッファ

メモリ共有

VM モードでは EL をリセットしない限り、外部 RAM の内容はクリアされません(内部 RAM はクリアされます)。この特徴をつかって、複数のプログラムや同一プログラムの複数 回の呼び出し(同一プログラムの複数インスタンス)でメモリを共有できます。

例えば、EL のプログラムが 250 バイト以上のデータを受け取る方法として、外部 RAM に 500 バイトのバッファを配置して、200 バイト、200 バイト,100 バイトの 3 回に分けて、 この領域にデータを書き込むことができます。外部 RAM は呼び出し毎にクリアされないた め、3 回分のデータを保持しつづけます。

#include "ses\_v3.h"
typedef struct
{
unsigned short offset;

```
unsigned char len;
unsigned char buff[1];
}
IO_PACKAGE;
DEFINE_AT(unsigned char, big_buff[512], 0x400, RAM_EXT);
IO_PACKAGE *input = NULL;
void main()
{
  input = (IO_PACKAGE *)pbInBuff;
  LE16 TO CC(&input->len);
  if (input->len != 0)
  {
          memcpy(big_buff + input->offset, input->buff, input->len);
         _exit();
  }
/* now got enough data and you can add operations here... */
_exit();
}
```

このコードでは2つのマクロを使っています。

DEFINE\_AT 指定 RAM の指定アドレスに変数を定義
 LE16\_TO\_CC リドルエンディアンから EL の数値フォーマットに変換

unsigned char の 512 バイト配列変数を外部 RAM に配置しています。呼び出し側が指定した配列内での書込み位置、長さ、データを使ってこの配列にデータをセットします。

コンピュータ側では Offset, Len, buff を適切にセットして、この EL プログラムを呼び出 します。

- 1. Offset = 0, len = 200, buff[200]
- 2. Offset = 200, len = 200, buff[200]
- 3. Offset = 400, len = 100, buff[100]

3回目の呼び出し後、big\_buff にはコンピュータ側から送られてきた 500 バイトのデー タがセットされています。

コードメモリ

これまで説明したメモリとは別に、読み出し専用のコードメモリというものがあります。初

期化済の大きな変数を保持するために利用できます。他のメモリと同じように使うことが できます。

code char message[1024] = "This should be a long message...";

内臓 OS 提供関数(SES 関数) でコードメモリに対して操作を行うものはありません。コ ードメモリから RAM メモリにコピーは標準の C 関数(strcpy(), memcopy()) をご 利用ください。

#### アライメントの問題

コンピュータと EL 間で構造体を受け渡しするにはバイト配置(アライメント)に配慮して ください。PC のコンパイラは効率よく処理するため構造体を機械語命令のサイズに合わせ て配置します。

例えば、PC コンパイラで構造体

struct{

char c; long l;

} x;

をコンパイルすると構造体のサイズは char が1バイト、long が4バイトだとして、合計 5 バイト、にはなるわけではありません。機械語命令サイズが4バイトだとすると、long は4バイト目に配置(アライン)され、結果4バイト+4バイトで8バイト構造体になりま す。一方、EL で同じ構造体を定義するとサイズは5バイトになります。PC 側と EL 側では 構造体のアライメントが異なるため、構造体をそのままの形で受け渡すことはできません。

PCのコンパイラでは構造体のアライメントをバイト単位になるように指定するオプション が備わっています。Visual Studio では、*#pragma pack (push, 1)*を指定できます。

このような形で構造体を宣言して EL 側と構造体のアライメントを一致させてデータの受け渡しができるようにしてください。

#### ビックエンディアン と リトルエンディアン

ビックエンディアンは、データを上位バイトから順にメモリに順に配置します。 0x12345678(16進数)というデータは、メモリ上に 12 34 56 78 と配置されます。

リトルエンディアンはデータを下位バイトから順にメモリに順に配置します。 0x12345678(16進数)というデータは、メモリ上に 78 56 34 12と配置されます。

Intel CPU を使ったコンピュータはリトリエンディアン方式です。一方、Keil C51 はビッ グエンディアン方式を採用しています。PC 側の数値を EL プログラムにそのままの形で渡 すと、受け取り側の EL プログラムは誤った値を受け取ってしまいます。例えば、PC(Intel CPU)の 0x12345678 はメモリ上では 78 56 34 12 となって EL プログラム に渡され ます。ビッグエンディアンとして数値を解釈する EL プログラムは、0x78563412 として データを受け取ります。正しく複数バイトの数値(short, int, long,float 等)を受け渡し するには、どちらかで変換しなければなりません。

開発キットには変換マクロが複数用意されています。LE16\_TO\_CC でリトリエンティア ン 16 ビット数値を LE 側のフォーマットに変換できます。CC\_TO\_BE16 は EL 側フォー マットの 16 ビット数値をビッグエンディアンに変換します。 CC は EL で使っている C Computer、LE はリトルエンディアン、BE はビッグエンディアンを表します。

倍精度浮動小数点数はコンパイラがサポートする標準データタイプではありません。SDK では倍精度浮動小数点数を表す DOUBLE\_T という構造体が用意されています。このデータ 型はどんなコンパイラを用いた場合でもリトリエンディアン形式になります。

データ型

型	char	integer	Short	Long	Float	double
長さ	1	2	2	4	4	8

## ライブラリファイル

プログラムのメモリモデルに応じて3つのライブラリが用意されています。

Large	ses51I.lib
Small	ses51s.lib
Compact	ses51c.lib

EL プログラムは Large Model で作成してください。ライブラリファイルは最初にプロジェクトフォルダにコピーして、それからプロジェクトに追加してください。

## EL プログラム作成時の留意点

- 1. スタックオーバフローを避けるため、多くのローカル変数(特に配列と構造体)を 使わない
- 2. 多くのメモリを使う配列と構造体の変数はできるだけ少なくする
- 3. 深い関数呼び出しのネスティングは避ける
- 4. 外部 RAM に変数を置く
- 5. 構造体のアライメントと CPU アーキテクチャ(ビッグエンディアンとリトルエン ディアン)に注意する
- 6. コメントは英語で(マルチバイト言語をコンパイラはサポートしていない)
- 7. sprint, sscanf の使用は避ける

## コンピュータ側からの EL へのアクセス

## 1. 一般アクセス(無認証アクセス)

一般アクセスとは認証無しで EL にアクセスすることを指します。一般アクセス権限で可能な操作は以下の通りです。

- 1. EL デバイスの検索 / 列挙、
- 2. デバイスへの接続(オープン)
- 3. ディレクトリ変更、
- 4. コントールコード送信、
- 5. デバイス切断 (クローズ)

```
無認証アクセスで EL を操作するサンプルプログラム
```

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "sense4.h"
int main(int argc, char** argv)
{
       SENSE4_CONTEXT s4ctx = {0}; /* current device context */
       SENSE4_CONTEXT *ps4ctx = NULL; /* for device context list */
       unsigned long ctx_size = 0; /* size of device context list */
       unsigned long ret = 0; /* return value */
       unsigned long len = 0; /* for returned data length. */
       /*step 1: Enumerate all the EL devices connected. */
       ret = S4Enum(NULL, &ctx size);
       if (ret != S4_SUCCESS && ret != S4_INSUFICIENT_BUFFER)
       {
               printf("Enumerate EL failed!<error code = %08x>¥n", ret);
               return 1:
       if (ctx size == 0)
       {
               printf("No EL found!¥n");
               return 1;
       }
       ps4ctx = (SENSE4 CONTEXT *)malloc(ctx size);
       ret = S4Enum(ps4ctx, &ctx_size);
       if (ret != S4_SUCCESS) /* Some error occurred. */
       {
               printf("Enumerate EL failed!<error code = %08x>¥n", ret);
               free(ps4ctx);
               ps4ctx = NULL;
```

```
return 1;
```

}

```
/* step 2: Open one of the device. For example, the first one. */
memcpy(&s4ctx, ps4ctx, sizeof(SENSE4_CONTEXT));
```

```
/* Here we can get device's Global Serial Number via s4ctx.bID. */
free(ps4ctx);
```

```
/* Or free it later if you want to access other devices.*/
ps4ctx = NULL;
ret = S4Open(\&s4ctx);
if (ret != S4 SUCCESS)
{
        printf("Open first EL failed!<error code = %08x>¥n", ret);
        return 1;
}
/* step 3: Now we can send some control code to device, for example,
control the LED. This step is not necessary. */
ret = S4Control(&s4ctx, S4_LED_UP, NULL, 0, NULL, 0, &len);
if (ret != S4_SUCCESS)
{
        printf("Light LED failed!<error code = %08x>¥n", ret);
        S4Close(&s4ctx);
        return 1;
}
/* step 4: We may change the current directory here. */
ret = S4ChangeDir(&s4ctx, "¥¥"); /* Change to root dir. */
if (ret != S4 SUCCESS)
{
        printf("Change to root dir failed!<error code = %08x>¥n", ret);
        S4Close(&s4ctx);
        return 1;
}
/* step 5: Do some more actions here... */
/* step 6: Close device. */
S4Close(&s4ctx);
return 0;
```

```
}
```

検索/列挙	S4Enum	通常、2度呼び出します。
		1 度目の呼び出しでは、第 1 引数を

		NULL, 第2引数を0にします。接続し
		ているすべての EL を列挙して、列挙に
		必要なデバイスコンテキストのサイズ
		を返します。例えば 5 つの EL が接続さ
		れていたら、5 つ分のデバイスコンテキ
		ストのサイズが返されます。いくつの
		EL が接続されているかは、返されたデ
		バイスコンテキストのサイズを、1つの
		デバイスコンテキストサイズで除算し
		ます。
		ctx_size/sizeof(SENSE4_CONTEXT)
		2度目の呼び出しては、接続しているす
		べての EL のデバイスコンテキストを保
		存するために必要分のメモリーを確保
		してから、第1引数に渡します。
		デバイスコンテキストにはシリアル番
		号が含まれます。これを使ってデイバス
		を識別できます。
		コンピュータ側から EL を操作する API
		では第1引数に対象 EL のデバイスコン
		テキストを指定します。
デバイスのオープン	S4Open	対象デバイスのデバイスコンテキスト
		を渡します。
コントールコード送出	S4Control	対象デバイスにコントールコードを送
		出します。LED 点滅/消灯やハードウェ
		アシリアル番号取得など各種操作を行
		えます。
ディレクトリ変更	S4ChangeDir	オープン直後、カレントディレクトリは
		ルートディレクトリです。
デバイスのクローズ	S4Close	処理終了後にクローズします。

EL 操作の一般的な手順は、対象となる EL デバイスを列挙後、オープンします。対象ディ

レクトリに変更してから、ユーザ PIN でログインしてプログラムを実行します。最後にデ バイスをクローズします。HID モードではプログラムがオープンしているデバイスを別の プログラムがオープンすることはできません。

対象 EL に接続(オープン)する処理は頻繁に必要になります。1つの呼び出しで済むよう にしておくと便利です。

{

```
unsigned long OpenDdongle(SENSE4_CONTEXT *ctx, int index )
       SENSE4_CONTEXT* ps4ctx = NULL;
       unsigned long ctx_size = 0; /* size of device context list */
       unsigned long ret = 0; /* return value */
       if( ctx == NUL )
               return S4_INVALID_PARAMETER;
       /*enumerate all the EL devices connected. */
       ret = S4Enum(NULL, &ctx_size);
       if (ret != S4 SUCCESS && ret != S4 INSUFICIENT BUFFER)
       {
               return ret;
       }
       if (ctx size == 0)
       {
               return S4_KEY_REMOVED;
       }
       ps4ctx = (SENSE4 CONTEXT *)malloc(ctx size);
       ret = S4Enum(ps4ctx, &ctx size);
       if (ret != S4_SUCCESS) /* Some error occurred. */
       {
               free(ps4ctx);
               ps4ctx = NULL;
               return ret;
       }
       if( index >(int)(ctx_size/sizeof(SENSE4_CONTEXT) -1 ))
               return S4_KEY_REMOVED;
       memcpy( ctx, ps4ctx+index, sizeof(SENSE4_CONTEXT));
       /* Here we can get device's Global Serial Number via s4ctx.bID. */
       free(ps4ctx);
```

```
/* Or free it later if you want to access other devices.*/
ps4ctx = NULL;
```

```
ret = S4Open(\&s4ctx);
```

return ret;

}

## 2. 開発者レベルアクセス

開発者 PIN でログイン(認証)後、開発者レベル権限で EL を操作ができるようになりま す。

- 1. ディレクトリの作成 / クリア / 削除
- 2. ファイルの作成/変更/削除

などが開発者権限を必要とする操作です。ログインは S4VerifyPin に開発者 PIN を与えて 呼び出します。成功すると開発者権限を取得します。

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "sense4.h"
#include "Psense4.h"
int main(int argc, char **argv)
{
       SENSE4 CONTEXT s4ctx = \{0\};
       unsigned char fid_exe[] = "d001"; // exe file ID
       unsigned long exe_size = 2048; // create file size
       unsigned char fid_dat[] = "d002"; // data file ID
       unsigned long dat size = 1024; // create file size
       char exe_path[] = "c: ¥¥s4demo¥¥demo1.hex"; // exf file path in disk
       char dat_path[] = "c: ¥¥s4demo¥¥data1.dat"; // data file path in disk
       unsigned char default dev pin[] = "123456781234567812345678";
       unsigned char old_dev_pin[] = "123456781234567812345678";
       unsigned char new_dev_pin[] = "876543218765432187654321";
       unsigned long len = 0;
       unsigned ret = 0;
       /* Open first EL if exists. */
       ret = OpenDongle(&s4ctx, 0);
       if (ret != S4_SUCCESS)
       {
               printf("Open EL failed! <error code = 0x\%08x> ¥n", ret);
               return 1;
       }
       /* Check whether a root dir exists. */
```

```
ret = S4ChangeDir(&s4ctx, "¥¥");
if (ret != S4_FILE_NOT_FOUND && ret != S4_SUCCESS)
{
     printf("Change to root dir failed! < error code =0x\%08x>¥n", ret);
     S4Close(&s4ctx);
     return 1;
}
/* If a root dir exists. */
if (ret != S4_FILE_NOT_FOUND)
{
       /* Verify Developer PIN to get full access privilege. */
       ret = S4VerifyPin(&s4ctx, old_dev_pin, 24, S4_DEV_PIN);
       if (ret != S4_SUCCESS)
       {
         printf("Verify dev PIN failed! <error code = 0x%08x>¥n", ret);
         S4Close(&s4ctx);
         return 1;
       }
       /* Delete old root dir. */
       ret = S4EraseDir(&s4ctx, NULL);
       if (ret != S4_SUCCESS)
       {
         printf("Delete root dir failed! < error code = 0x\%08x > 4n", ret);
         S4Close(&s4ctx);
         return 1;
       }
}
/* Create new root dir. */
ret = S4CreateDir(&s4ctx, "¥¥", 0, S4_CREATE_ROOT_DIR);
if (ret != S4_SUCCESS)
{
       printf("Create new root failed! <error code = 0x%08x>¥n", ret);
       S4Close(&s4ctx);
       return 1;
}
/* Verify Developer PIN to get full access privilege. */
ret = S4VerifyPin(&s4ctx, default_dev_pin, 24, S4_DEV_PIN);
if (ret != S4_SUCCESS)
{
       printf("Verify dev PIN failed! <error code = 0x%08x>¥n", ret);
       S4Close(&s4ctx);
       return 1;
}
/* Write disk file to EL. */
ret = PS4WriteFile(&s4ctx, fid_exe, exe_path, &exe_size,
               S4_CREATE_NEW, S4_HEX_FILE, &len);
```

```
if (ret != S4_SUCCESS)
{
       printf("Write exe file failed! <error code = 0x\%08x>¥n", ret);
       S4Close(&s4ctx);
       return 1;
}
ret = PS4WriteFile(&s4ctx, fid_dat, dat_path, &dat_size,
               S4_CREATE_NEW, S4_DATA_FILE, &len);
if (ret != S4_SUCCESS)
{
       printf("Write data file failed! < error code = 0x\%08x>¥n", ret);
       S4Close(&s4ctx);
       return 1;
}
/* Change Developer PIN. */
ret = S4ChangePin(&s4ctx, default_dev_pin, 24,
                       new dev pin, 24, S4 DEV PIN);
if (ret != S4 SUCCESS)
{
       printf("Change dev PIN failed! <error code = 0x%08x>¥n", ret);
       S4Close(&s4ctx);
       return 1;
}
/* Close EL. */
S4Close(&s4ctx);
/* everything done! */
printf("Congratulations!¥n");
return 0;
```

}

\*このコードは問題ないように見えますが、1つ大きな欠陥があります。S4VerifyPin で 開発者権限を取得すると、デバイスをS4Closeで閉じても権限状態はそのままになります。 S4Closeは認証状態をクリアしません。そのため、次にS4Openでデバイスをオープンする と開発者レベル権限のままなので、開発者権限操作が可能です。認証状態をクリアするには S4Controlでコントロールコード S4\_RESET\_DEVICE を送出してください。

開発者 PIN は外部に絶対に漏れないようにします。開発者側が EL の設定を行うコードで は使っても、ユーザに渡すプログラム内で絶対に使わないようにしてください。

## 3. ユーザレベルアクセス

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "sense4.h"
#include "Psense4.h"
int main(int argc, char **argv)
{
       SENSE4_CONTEXT s4ctx = \{0\};
       unsigned char fid_exe[] = "d001"; // exe file ID
       unsigned char user_pin[] = "12345678";
       unsigned long len = 0;
       unsigned ret = 0;
       unsigned char input[256] = \{0\};
       unsigned long input len = 128;
       unsigned char output[256] = \{0\};
       unsigned long output_len = 251;
       /* Open first EL if exists. */
       ret = OpenDongle(\&s4ctx, 0);
       if (ret != S4 SUCCESS)
       {
               printf("Open EL failed! <error code = 0x%08x>¥n", ret);
               return 1;
       }
       /* Change to a root dir. */
       ret = S4ChangeDir(&s4ctx, "¥¥");
       if (ret != S4_FILE_NOT_FOUND && ret != S4_SUCCESS)
       {
           printf("Change to root dir failed! < error code =0x\%08x>¥n", ret);
           S4Close(&s4ctx);
           return 1;
       }
       /* Verify Developer PIN to get full access privilege. */
       ret = S4VerifyPin(&s4ctx, user_pin, 8, S4_USER_PIN);
       if (ret != S4_SUCCESS)
       {
               printf("Verify User PIN failed! <error code = 0x\%08x>¥n", ret);
               S4Close(&s4ctx);
               return 1;
       }
       /* Invoke exf 0xd001. */
       ret = S4Execute(&s4ctx, "d001", input, input_len,
                         output, output_len, &len);
```

このアクセスレベルの目的は、EL 内部の実行ファイル(EXF)を呼び出すことです。

```
if (ret != S4_SUCCESS)
{
          printf("Invoke 0xd001 failed! <error code = 0x%08x>¥n", ret);
          S4Close(&s4ctx);
          return 1;
}
/* Close EL. */
S4Close(&s4ctx);
// Reset Device if necessary
/* everything done! */
printf("Congratulations!¥n");
```

return 0;



Keil C51 の使い方

## Keil C51 の特徴

Code Mode	VM Mode
Code bit	8 bits
Byte sequence	BE(Big-Endian)
<b>Optimization Efficiency</b>	High
Memory Structure	Internal RAM + External RAM
	Commercial software. Special
Notes	attention shall be given to the issue
	of byte sequence.

ビルドの結果は HEX ファイルになります。このままでは EL では実行できません。コー ドポートする前に、hexbin.exe で BIN (バイナリ)フォーマットに変換してください。付 属の DevTest.exe は HEX ファイルが指定されると、BIN ファイルに自動変換します。

## EL 開発用パッチ

Keil C51 をインストールしたら、IDE¥keil ディレクトリにある wizard.exe を実行して ください。Keil で EL プログラム開発に必要なセットアップを行います。

## Keil C51 O IDE - uVision

プロジェクト作成

- 1. Project->New Project でプロジェクト名を設定して、保存します。
- 2. Wizard.exe でパッチを導入済みであれば、"Select a CPU Data Base File"ウィン ドウが表示されます。"Senselock Device" を選択してください。

Senselack Devices	•
0K	Cancel

Select Device for Target 'Target 1'			2	3
CPU				
Vendor: Senselock				
Device:		Use Extended Linker (LX51) instead of BL51		
Family:		Use Extended Assembler (AX51) instead of A51		
Data base	Description:			
SenseLock EL V2.3.2			*	
	OK	Cancel		

Data base リストで Senselock->SenseLock EL V2.3.2 を選択してください。

 パッチを導入していなければ、Data base リストボックスで CPU として Intel->8052AH を選択してください。



\*このダイアログは、Project->Select Device for Target 'Target1'で表示でき ます。

4. プロジェクトのオプション設定

IDE の左リストで Target を選択してから、Project->Options for Target 'Target 1' でオプションダイアログが表示させます。

- a. Target タブでMemory Model を Large に設定。
- b. Output タブで Create HEX File をチェック

Options for Target 'Target 1'
Device Target   Output   Listing   C51   A51   BL51 Locate   BL51 Misc   Debug   Utilities
Xtal (NHz): 20.0 Henory Medel: Large: variables in XDATA Code Rom Size: Small: variables in DATA Operating Compact: variables in PDATA
Off-chip Code memory Start: Size: Eprom E
Code Banking     Start:     End:       Banks:     2 y     Bank Area:     Du0000     Du0000
OK Cancel Defeults
Options for Target 1'
Device   Target Output   Listing   C51   A51   BL51 Locate   BL51 Misc   Debug   Utilities
elect Folder for Objects Name of Executable: test
@ Create Executable: . \test
✓ Debug Informatio
Create HEX Fi: HEX HEX-80
← Create Library: .\test.LIB
After Hake
🔽 Beep When Complete 🧮 Start Debugging
Eun User Program #1 Browse
Eun User Program #2
OV Carrel Defeater

4. ヘッダファイルとライブラリファイルの追加

Ses\_v3.h をプロジェクトディレクトリか、C51の INC ディレクトリにコピーしてくだ

さい。また、**Project**->**Components, Environment, Books** でライブラリファイ ル*ses51L.lib*をプロジェクトに追加してください。ライブラリは開発キットの IDE/keil/lib 内にあります。

7. C のソースコードファイルを追加してください。

## プログラムのビルトとデバッグ

**Project**->**Build target** でプログラムをビルドします。成功したらプロジェクトディレクトリに HEX ファイルが出来上がります。

## <u>デバッグするための設定</u>

デバッグにはソフトウェアシミュレータを利用できます(uVisionのバージョンによって は利用できない可能性があります)。パッチ導入時にシミュレータはセットアップされます。 また、手動での設定手順は以下の通りです。

- a. 開発キットの IDE/Keil/bin 内の S4Situmulator.dll と vfsView.exe を Keilの BIN ディレクトリ(...¥Keil¥C51¥BIN¥) にコピー
- b. 設定ファイル( ...¥Keil¥Tools.ini )の [C51] セクションに

## TDRV0=BIN¥S4Simulator.dll("SenseLock EL Simulator")

を追加。既に TDRVO という名前で DLLが登録されていたら、まだ使われて いない番号に変更

TOOLS - Notepad	×
<u>File Edit Format View Help</u>	
<pre>[C51] 800K0-HLP\RELEASE_NOTES.HTM("Release Notes") 800K1-HLP\GS51.PDF("uvision2 Getting Started") 800K2=HLP\C51.PDF("C51 user's Guide")</pre>	^
BOOK3=HLP\C51LIB.CHM("C51 L1brary Functions",C) BOOK4=HLP\A51.PDF("Assembler/Utilities")	
BOOKS-HLP\TRS1.CHM("RTXS1 Tiny User's Guide") BOOK6-HLP\DBG51.CHM("uVision2 Debug Commands")	
BOOK7-HLP\ISD51.CHM("ISD51 In System Debugger") BOOK8-HLP\FlashMon51.CHM("Flash Monitor") BOOK8-HLP\FlashMon50.HTM("Mon390: pallas contiguous Mode Monitor") TDRV0-BIN\Mon51.DLL ("Keil Monitor-51 Driver")	Ŧ
TDRVI-BIN\ISDS1.DLL ("Keil ISDS1 In-System Debugger") TDRV2-BIN\MON390.DLL ("MON390: Dallas Contiguous Mode") TDRV3-BIN\LPCZEMP.DLL ("LPC900 EPM Emulator/Programmer") TDRV4-BIN\ULZUPSD.DLL ("ST-uPSD ULINK Driver")	
RTOS1=RTXTINY.DLL ("RTX-51 TINY") RTOS2=RTX51.DLL ("RTX-51 Full")	
Version=V7.0 PATH="C:\Keil\C51"	
SN-K10ZP-SIUSH-A01UE TDRV5-BIN\54Simulator.dll("SenseLock EL Simulator")	
	Ŧ
< m +	

 c. uVision を再起動後、Project->Options for Target 'Target 1' で表示 されるダイアログで Debug タブを開き、SenseLock EL Simulator を選 択。

Options for Target 1'											
Device Target Output Listing C51 A51 BL51 List	ocate BL51 Misc Debug Utilities										
Use Simulator Settings	C Use: SenseLock EL Smulator ▼ Settings										
I Load Application at Startup  Go till main()	✓ Load Application at Startup										
Initialization File:	Initialization File:										
Edi	L. Eck.										
Restore Debug Session Settings Breakpoints P Toolbox Watchpoints & PA Memory Display	Restore Debug Session Settings Breakpoints Watchpoints Memory Display										
CPU DLL: Parameter:	Driver DLL: Parameter:										
Dialog DLL: Parameter:	Dialog DLL: Parameter:										
OK Ca	ncel Defaults										

d. OK ボタンでダイアログを閉じて完了です。

## <u>デバッグ操作</u>

**Debug->Start/Stop Debug Session** メニュを選択。EL用のデバッグ設定が正しく されていれば、Input ウィンドウが開きます。

vFS file paths: C:\Users\Leo\Desktop\vFileSys.dat													
New Modify Default ExF: none													
Input data length:         250         Bytes           00000         0													
Load	Load Save As OK Can												

このウィンドウ上半分は EL ファイルシステムのシミュレーション設定、下半分はコミュ ニケーションバッファの内容表示になっています。

## <u>ファイルシステムのシミュレーション</u>

EL ファイルシステムはプロジェクトディレクトリ内の vFileSys.dat というファイルをつ かってシミュレーションされます。EXF が EL のファイルシステム関数を呼び出してい て、ファイルシミュレーションを利用したければ New ボタンで vFileSys.dat を作成し てください。

Modify ボタンでルートディレクトリに対する操作を行えます。シミュレートされたファ イルシステムでは ROOT ディレクトリのみ利用できます。サブディレクトリは作成でき ません。

🕵 vFileSys - vfsView				
File Edit View Help				
🖉 🖬 🕹 🖻 💼 💡				
Name	Length	Property	Status	
🚞 ROOT	0x10000			
Ready			NUM	1 1/2

ROOT ディレクトリには4種類のファイルを作成できます。すべて読み書き許可になります。

- 1. 実行ファイル(EXF)
- 2. データファイル(DAT)
- 3. RSA 公開鍵ファイル(既定サイズ 136 バイト)
- 4. RSA 秘密鍵ファイル(既定サイズ 330 バイト)

ファイルを作成するには、ROOTを選択して、右クリックで表示されるメニュで New File を選択します。

🛃 vFileSy	s - vfsView					
File Edit	t View Help					
🛸 🖬	ራ 🖻 💼 🥊					
Nane		Lengt	h	Property	Status	
💼 🔣	OT	0x1000	00			
	New File					
	Erase					
Ready					NUM	

作成するファイルを指定して OK で作成が完了します。

Create New File											
Name: EF02 Type: DAT	Length: 128										
☐ R/W Properties	ОК	Cancel									

作成したファイルを編集するには、ファイルを選択して右クリックで表示されるコ ンテキストメニュで Edit を選択します。

🕵 vFileSys - vfsView										
File Edit View Help										
🚅 🖬   🌡 🖻 💼   🦹										
Nane	Length	Length Property Status								
E- 📄 ROOT	0x10000									
EF02	Edit	R/V/-								
Ready										

**16** 進データを直接編集、ディスクファイルからインポート、ディスクファイルヘエ クスポートなどの操作が可能です。

Bir	n	aŋ	/ E(	dit	t Di	ial	log														х	-
			)0( )1( )2( )3( )5( )6( )7(					000 000 000 000	00 00 00 FA 00 00 00	00 00 DF 00 CF 00 00 00	00 AD 00 DF 00 00 00	00 FA 00 DF 00 00	00 DF 00 DF AD 00 00	00 00 00 EE FÅ 00 00	00 00 00 00 EE DF 00 00	00 00 00 EE EA 00 00	00 00 00 E0 DF 00 00	00 00 00 00 EE 00 00		000000000000000000000000000000000000000		
				L	oad	ł				Sav	e A	5			0	ĸ			Can	cel		

EXF プログラムによる自身のプログラムファイルに対する操作

実行ファイル EXF は、SES 関数 \_create で自身の EXF ファイルをオープンすることが できます。しかし、シミュレートされたファイルシステムには、実行中プログラムの EXF は保存されませんので、そのような \_create 呼び出しは失敗します。この処理をシミュ レートするには、ファイルシステムに実行ファイルを作成してデフォルトファイルとして 設定します( EXF を選択後、右クリックメニュで Set To Default を選択)。EXF が自身のファイルをオープンすると、このデフォルトファイルが開かれます。

🙀 vFileSys - vfsView						
File Edit View Help						
🛎 🖬   X 🖻 🛍   🎗	2					
Name	Length	Length Property Stat				
E- COT	0x10000					
	0800x0	R/V/-				
EF03 Ed	it	- <i></i>	Enable			
Ready	t to Default		NUM			

<u>コミュニケーションバッファ</u>

シミュレータは EL のコミュニケーションバッファをシミュレートします。コンピュータ 側プログラムから EL に転送されるデータはシミュレータのコミュニケーションバッファ にコピーされます。このコミュニケーションバッファ内容は手動で変更可能です。

1. コミュニケーションバッファウィンドウのアドレスを右クリック

2. ポップアップメニュで選択したアドレスに挿入するデータを指定

Input(Release Candidate Version)			
vFS file paths:			-
New Modify Default ExE: Dope			
Input data length:	250 Bytes		
0000 00 00 0 0 0010 00 00 0 0020 00 00 0 0030 00 00 0	Copy Paste	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	
0040 00 00 0 0050 00 00 0 0070 00 00 0 0070 00 00 0 0080 00 00 0 0090 00 00 0	Insert String Insert BYTE(Decimal) Insert WORD(Decimal) Insert DWORD(Decimal)	$\begin{array}{cccccccccccccccccccccccccccccccccccc$	00 00 00 00 00 00 00
Load	Insert Float Insert Double	Can	cel
	Insert Form File		

## デバッグ実行

ファイルシステムとコミュニケーションバッファを設定したら、[OK]ボタンでデバッグ処理を開始します。以降 Keil 側の処理になります。詳細は Keil のマニュアルをご参照ください。

## 実行エラー

デバッグ中に実行エラーのダイアログが表示されたら、エラコードを確認してください。また、**View->Project Window** で表示される Project Workspace ウィンドウの Regs タブで表示される内容も確認してみてください。このタブには CPU の各種レジスタ値が 表示されます。

R0~R7 (Universal Registers),

R0~R1 (IRAM, Indirect Addressing Registers of Internal Memory),dptr (XRAM, Indirect Addressing Registers of External Memory),sp (Stack Pointer),

PC (Program Pointer).

Project Workspace		
Register	Value	
E-Begs		
r0	0x00	
ri	0x00	
r2	0x00	
r3	0x00	
r4	0x00	
r5	0x00	
r6	0x00	
Tr	0x00	
⊡ Sys		
a	0x00	
Ъ	0x00	
sp	0x00	
dptr	0x0000	
PC \$	0x0000	
+ psv	0x00	
🖹 Files 🗮	🛄 B   🍕 Fu   🦏 Te	

一般的に遭遇するエラー:

dptr が範囲外 (許容範囲は 0x0000 から 0x08ff)

SP オーバフロー(許容範囲は 0x07 から 0xff、スタックにデータが積まれると ポインタ値は減ります)

PC が範囲外 (コード範囲外を指している可能性があります)

## デバッグ終了

メニュー **Debug->Start/Stop Debug Session** でデバッグ状態から抜けることが できます。

ELプログラムが SES関数 \_exit() を呼び出してプログラムが終了するとデバッグが終了 します。この時、ELプログラムが \_set\_response() でセットしたデータを表示するダイ アログが現れます。 付属ツール

## HEX ファイル変換ツール

Keil でプログラムをビルトすると HEX 形式の実行ファイルが出来上がります。この形式 のファイルを EL は実行できません。転送する前に BIN ファイルにしなければなりません。

SDK 内に hexbin.exe というコマンドラインユーティリティが付属します。このユーティリティで HEX ファイルから BIN ファイルに変換できます。

➢ Hexbin.exe [HEX ファイル] [BIN ファイル]

API と付属のツールは HEX が指定されると、内部で HEX ファイルから BIN ファイルに 変換します。例えば S4WriteFile API にファイルタイプとして S4\_HEX\_FILE を指定し て HEX ファイルを渡すと、BIN ファイルに変換してから EL に書き込みます。

## **EL** デバイス設定ツール (**DevTest.exe** )

tool フォルダ内の devtext.exe を使うと GUI で EL に対して各種操作を行えます。

DevTest.exe を起動するとコンピュータに接続している EL を検索します。もし、接続し ていればプログラムのメインウィンドウが表示されます。接続されていなければ Error ダ イアログが表示されてから、メインウィンドウが表示されます。





**Reconnect** でデバイスに接続して、接続したデバイスの情報を表示します。[Disconnect to device]ボタンで接続を解除します。



## 1. EL デバイスの初期化 / 再初期化

ELを初めて使う前にルートディレクトリを作成しなければなりません。また、既に利用している EL はルートディレクトリを削除すると初期化できます。DevTest では Recreate Root メニューでルートティレクトリの作成、削除が行えます。



最初のフィールド(Developer PIN) にはルートディレクトリの既定の開発者 PIN (\*123456781234567812345678")が自動入力されます。初めて EL にルートティ レクトリを作成する場合、Developer PIN フィールドは既定値から変更する必要はあり ません。Device ID には 8 バイトの任意の ID を設定します。この ID は EL 列挙 API が取得する EL 情報の一部としてセットされます。コンピュータ側のプログラムは、この ID でプログラムが操作対象とする EL デバイスかどうかを識別できます。

ルートディレクトリ作成済みの EL デバイスでは、ルートディレクトリの正しい開発者 PIN を設定してください。

ルートディレクトリが再作成されると、ルートディレクトリの開発者 PIN と ユーザ PIN は既定値にリセットされます。なお、このツールではサブディレクトリは作成できません。

## 2. ファイルのダウンロード

Download メニュでコンピュータ上のディスクファイルを EL に転送できます。

転送可能なファイル( Download the type of file コンボで指定 )

- 1. 実行ファイル(HEX/BIN)
- 2. データファイル
- 3. RSA 鍵ファイル(S4\_RSA\_PUBLIC\_KEY / S4\_RSA\_PRIVATE\_KEY 型)

🔀 Elite EL developm	nent and test tool
Menu	Download 🥸
Ø	
Recreate Root	Download the file path:
Download	C:\S4demo\demo1.bin browse
0	Download the name of subdirectory:
Emoty Directory	Download file name: ef01
PIN Management	File size(byte): 398
0 PC Keys	Download the type of file: VM executable file(BIN)
Reconnect	<ul> <li>It is read-writable for executable file</li> <li>Overwrite the original file</li> </ul>
	download file generate key pair

実行ファイルは HEX か BIN 形式のファイルのみを指定可能です。HEX 形式のファイル が指定されると BIN 形式に変換して転送します。変換後の BIN ファイルはカレントディ レクトリに保存されます。

RSA 鍵ファイルのフォーマットは S4\_RSA\_PUBLIC\_KEY か S4\_RSA\_PRIVATE\_KEY でなければなりません。このフォーマットの鍵は EL 内部の鍵の形式とは異なりますが、転送前に変換後、転送します。

転送するファイル名を指定する前にファイルタイプを選択してください。ファイルタイプ

選択後に転送ファイル名を設定すると、ファイルサイズが計算され File Size に自動的に 設定さます。この値は、ファイルが実際に EL に保存されるときのサイズです。手動でファ イルサイズを自動計算値より大きな値に変更しても構いません。指定サイズでファイルは 作成されます。

ディレクトリとファイル名は2バイトの16進数で指定します(ルートディレクトリは ¥を指定します)

指定ファイルが実行ファイルの時、It is read-writable for executable file チェックボッ クスをチェックすると、実行ファイル属性が read-writable になります。既定は read only です。

EL 内の既存ファイルを更新するには、Overwrite the original file チェックボックスを チェックしてください。転送するファイルのサイズは既存ファイルサイズより小さくなけ ればなりません。

## 3. 実行ファイルの起動

Execute メニュで EL 内の実行ファイルを起動することができます。User PIN フィール ドには既定値(12345678)が自動設定されます。ファイル名 (The name of)は絶 対パスで指定してください。実行ファイルへの引数は16進で指定します。

実行ファイルからの結果は Parameter of execution file フィールドに表示される
--

🚰 Elite EL development and test tool		
Menu	Execute	0
1		
Recreate Root	user PIN:	
<u> </u>		Execute
Download	The name of	
Ø	\ef01	
Execute	- Float - Stripp	
100		Float → String
Empty Directory		
SIN Massace		String -> Float
Fini Management	operating in XA mode	
Ø.	narrameter of execution file	
PC Neys	parameter or execution me	
<b>B</b>	123456	
Neconnect		

## 4. ディレクトリのクリア

Empty Directory で、指定ディレクトリをクリア(空にすることが)できます。クリア すると指定ディレクトリの 開発者/ユーザ PIN は既定値に戻ります。

🜠 Elite EL development and test tool		
Menu	Empty Directory	
🗭 Recreate Root	The name of directory which is about to be empty:	
🐼 Download	ef01	
<b>Frecute</b>	The developer PIN of the directory	
Empty Directory	After clearing the directory, its original content will be erased as well. In addition, all PINs under the directory will be reinitiated by default PIN, but the	
PIN Management	directory name will be reserved.	
PC Keys	Clear the directory	
Reconnect		

サブディレクトリをクリアしても、ディレクトリ内のファイルとディレクトリは削除され ますが、削除対象として指定したディレクトリは削除されません。PIN はリセットされま す。ディレクトリを削除するにはそのディレクトリを含む上のディレクトリをクリアして ください。

## 5. PIN の変更

PIN Management で指定ディレクトリの開発者 PIN とユーザ PIN を変更できます。

Ę	🕼 Elite EL development and test tool 🛛 🗖 📼		
	Menu	PIN Management	۵
	1	·	
		directory name:	
	3	N	-
	Download	Diginal DEV PIN:	
	۵		-
		1	
	<b>1</b>	New DEV PIN:	
	Empty Directory		
	\$	Confirm:	
	PIN Management		-
	ø		
	PC Keys	modify the type of PIN:	_
	Ð	DEV PIN	·
	Reconnect	Modify DEV PIN	

## 6. RSA 鍵の生成

PC Keys メニュで RSA 鍵ペアをコンピュータ側で生成できます。 生成される鍵は S4\_RSA\_PUBLIC\_KEY、S4\_RSA\_PRIVATE\_KEY フォーマットになって います。

E	🕼 Elite EL development and test tool			
	Menu	PC Keys 🔞		
	🗭 Recreate Root	input the directory contains generationfile:		
	S Download	C:\S4demo Browse		
	<b>Ø</b> Execute	input the public key name: pub.pub		
	🔊 Empty Directory	input the private key name: pri.pri		
	PIN Management	It is available to generate RSA public-private key pair according to PKCS standard, which could be imported to the device by the tool.		
	1 Reconnect	generate the key pair		

生成した鍵ペアを保存するディレクトリとファイル名を指定してください。