TOPPERS 活用アイデア・アプリケーション開発コンテスト

部門	：	活用アイデア部門 アプリケーション開発部門 がじぇるね IoT 部門
作品のタイトル	：	athrill+TOPPERS/SSP(シュリンク版)による 初級者向けハンズオンセミナー教材の作成
作成者	：	アライブビジョンソフトウェア株式会社　代表　高橋和浩
共同作業者	：	
対象者	：	プログラムを少し書ける程度の初級者 RTOS に触れてみたいが時間が無い方 RTOS の仕組みに興味ある中級者
使用する開発成果物	：	仮想環境　athrill / TOPPERS/SSP カーネル

目的・狙い

昨年度、Raspberry Pi による OS セミナ教材を作成、実際に島根県の公的機関にてセミナーを実施しました。 やってみて思ったのは実機で行うのはある程度のスキルが必要なことと、準備に時間を要してしまうデメリットがあります。
今回、仮想環境を使うことで初級者でもさらっと自分で学習できたり、短時間のハンズオンセミナができる教材もあれば役に立つのではと思いました。
SWEST での登壇させていただいたときも、Raspberry Pi の USB シリアルのケーブルや開発環境など大半がセットアップに要してしまうことや、Raspberry Pi が無いとまず参加できないなどのデメリットがあったと思います。
これらを解決すべく、比較的初級者向けに広報できるようなハンズオンセミナ教材を作成したいと考えました。

アイデア/アプリケーションの概要

以下 3 つで構成されます。

1. ハンズオンセミナ教材

ハンズオンでの 1 時間セミナさらには 1 日程度のセミナを想定したセミナー教材です。 1 時間ものは、パソコンが利用できるならだれでも athrill で RTOS 動作まで確認できる内容になります。RTOS を簡単に触れてみたいということに対応できるかと思います。

一方、1 日のセミナでは、リアルタイム OS のディスパッチャの実装をタイマ割り込みサンプルプログラムからマイコン依存部の実装して動作させるまでを講義する内容で考えています。自作 OS セミナの 1 日版になると考えています。

2.athrill (V850) 用 TOPPERS/SSP シュリンク版

教材で利用するプログラムです。非常に軽くデバッグ機能もある athrill 上で動作する TOPPERS/SSP シュリンク版を先日作成しました。 V850 は実際には使ったことはありませんでしたが、(RH850 は絶賛使用中ですが)マイコン仕様を調査済みであれば 1 日で移植できました。 C 言語ディスパッチャがベースなので、一部（割り込み許可禁止、Callee セーブレジスタ（非スクラッチレジスタ）の保存と復元、スタック切り替え、PSW の操作など)をインラインアセンブラで追加することで大方の移植が可能です。

ソースはここで公開しています。

https://github.com/alvstakahashi/TOPPERS_SSP_shrink_V850_For_athrill

2. 自主学習用ガイドブック

セミナに参加できなくても、パソコンさえあれば自主学習できるようなガイドブック化したものを作成します。 セミナ参加者の予習にもなり、より深く学ぶために効果的かと考えています。

そのほか 今後の展望など：

おそらく、Atrhill が RH850 など主要なマイコンの対応がされればより利用者の興味がわくことと期待しております。

ハンズオンセミナは、OSC 大阪や京都などのイベントや無料での出前セミナー等で実施できればと思っています。

アイデア/アプリケーションの詳細

経緯:

OSC 京都 2018 にて、TOPPERS ブースのお手伝いに行った際、同様にデモで athrill およびその作者の森さんに出会い　TOPPERS/SSP を Ahrill で動かしたいという話があり、シュリンク版ならすぐできますよという会話をしたのですが、athrill はなんだかおもろそうだと思い、まず、自分で V850 athrill に移植してみました。

ただ作ってみて、以前　Raspberri Pi のハンズオンセミナが結構手間だったことを考えると athrill を使ってハンズオンセミナならより簡単に C 言語覚えたての人からでもなにかやった感のある内容を短時間でできるように思いました。

各成果物の詳細(想定する内容)

１．athrill (V850) 用　TOPPERS/SSP シュリンク版

教材の元になるプログラムです。

ここに公開中。

https://github.com/alvstakahashi/TOPPERS_SSP_shrink_V850_For_athrill

また、完成品だけがアップされているのではなく、OS なしの athrill のベアメタルのプログラムをベースラインとして、自分が移植していくところが git の履歴でみることができるようになっています。

1) C 言語記述化ディスパッチャ

C 言語記述化ディスパッチャのポイントは、setjmp() longjmp()の標準関数および、ベアメタルでのタイマー割り込みハンドラを使用した遅延ディスパッチ部になります。

V850 用 GCC は ARM で利用できた attribute 属性による割り込みハンドラの記述ができないようなので､インラインアセンブラにてレジスタ保存/復帰を行うように実装しました。

主だった移植部分は、target_kernel.h に集約　主に以下のマクロを定義しています

```
#define         set_task_stack(x)               __asm__( "mov    %[Rs1],sp"::[Rs1]"r"(x))

#define disable_IRQ()               __asm__("di")
#define enable_IRQ()                __asm__("ei")

#define         ipl_maskClear()                 set_intpri(INTPRI_ENAALL)
//これは ASP3 の依存部の関数呼び出し
/*
 *  さらに非スクラッチレジスタを退避する
*/
```

```
#define saveCTX()      do { ¥
                       _asm__("add                -4 , sp;":::);¥
                       _asm__("st.w               r2 , 0[sp];":::);¥
                       __asm__("PREPARE
      {R20,R21,R22,R23,R24,R25,R26,R27,R28,R29},10;":::);¥
                                } while(0)

#define loadCTX()      do { ¥
                       __asm__("DISPOSE
      10,{R20,R21,R22,R23,R24,R25,R26,R27,R28,R29};":::);¥
                       __asm__("ld.w              0[sp] , r2;":::);¥
                       __asm__("add               4 , sp;":::);¥
                      } while(0)

/*
 *   コンテキストの参照
 *
 *   割込みの戻り先がタスクかどうかを判断するために intnest
 *   を使用している．これを用いてコンテキストを判断する．
 */

Inline bool_t sense_context( void )
{
/*
 *   割込み発生回数を保存する変数
 */
extern volatile   uint16_t        intnest;

      /*   ネストカウンタ 0 より大なら非タスクコンテキスト   */
      return ( intnest > 0U );
//    return 0;

}
#define        get_sp(status)    __asm__("mov     sp,%[Rd]":[Rd]"=r"(status));
```

さらにタイマー割り込みについては、athrill のサンプルとして移植されておる TOPPERS/ASP3 V850 用のソースを一部利用しています。

2. ハンズオンセミナ教材

1 時間用および 1 日用共通な部分として、開発ツールのインストールになります。
Windows 32bit 64bit の両方を対応するものとします。
インストールは、MinGW32bit および　V850 の GCC、さらに SAKURA エディタになります。
MinGW は基本的にインターネットからのインストールになりますが、Eclipse のオールインワンパッケージなら、Eclipse に付属した MINGW が解凍して Path を設定するのみになります。　V850 の GCC はユーザー登録が必要なので各自登録していただくことが前提ですが、これらの手順にしたがいインストールするだけです。
あとは、ターゲットである　athrill と　V850 版の TOPPERS/SSP をそれぞれ Make するだけで、構築が完了します。
一方
1 日用のセミナのカーネル移植の内容は、ベアメタルのタイマー割り込みのソースに
1) カーネル起動部、割り込みハンドラからのハンドラーコール
2) TOPPERS/SSP シュリンク版のカーネル(非依存部)を追加
3) Target_kernel.h chip_timer.c の作成追加
これを順次コーディングしていく内容でになり、OS 自作セミナの短縮版になります。

以上