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
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 「Visual Studio Code のリモート機能を使ったお手軽 TOPPERS サンプル」
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

TOPPERS 関連の RTOS のサンプルやサンプルを元にした教育セミナーで使うために実際のハードウェアを用意する必要がありますが、ハードウェアなしでも可能にできるとさらに視野が広がると考えています。athrill(V850 シミュレータ）を利用したサンプルプログラムですが、これを比較的短時間かつユーザーとともに実演できるセミナー教材にしてみました。また、2018 のアイデアから 2 年間に開発環境の進歩があり、Windows には　WSL2(Linux サブシステム 2)、Docker For Windows,Visual Studuo Code にはそれらに連携しシームレス化するリモートコンテナ機能が搭載されました。また、athrill の DockerHub には、ビルド環境まで構築されたイメージの公開がなされています。これらに、本 TOPPERS-SSP シュリンク版 athrill 用に手を加えたことにより、あたかも「IDE で HelloWorld 化した RTOS 実行環境」が実現できたと思っています。実演動画　3 分程度も添付していますのでそちらもご覧ください。

アイデア/アプリケーションの概要

github
https://github.com/alvstakahashi/TOPPERS_SSP_shrink_V850_For_athrill
に動画以外は公開しております。

１．ハンズオンセミナー当日用パワーポイント
「VSCode を利用した仮想化リアルタイム OS 実演.pptx」
添付しております。LT でも利用できる程度のセミナー用パワーポイントになります。数分で、実行できるデモの説明資料になります。

２．実施動画
Youtube https://t.co/OsBnVeu8tN?amp=1 にアップしました。
本コンテスト用に作成した動画になります。
上記 1 のキャプチャーしたものになります。

３．事前準備設定資料
「Athrill 実行環境事前設定の説明.docx」添付しております。
ハンズオンセミナーを想定した場合に、それまでに PC のセットアップをする手順を記載しています。WSL2 および Docker の設定なのです。
もっと最適でスマートな方法もあるかと思いますが、一旦これでセットアップできるものとなっています。

４．athrill 対応 TOPPERS-SSP シュリンク版
github で公開している athrill 用の TOPPERS-SSP シュリンク版になります。
Dockerfile を追加したことで、VScode にて Docker に接続することができます。

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。

アプリケーション/教材の対象者

RTOS の Hello World ということを考えており、C 言語がようやくわかった初級者やそういった方が RTOS に初めてふれてみたいという方を対象と考えています。
そうでなくても、環境構築が面倒とお考えの方にも利用価値があると思います。
ただし、本対象の環境は Windows 限定しております。
さらに Windows においても、バージョンおよび CPU 等少し限定させていただいています。
理由は、それが最も作業が少なく、もともと環境クリアされている人が大多数であろうという推測に基づくものです。
また、事前設定資料には比較的多くの設定を行っていますが、あくまで標準的な WSL2 と Docker のセットアップと考えており、本教材以外においてもプログラミング環境として応用できるものと考えています。今回環境を限定した理由については、Corei シリーズの Windows10 64bitPro さえ用意すれば可能なので、今であれば、テレワーク用など中古市場を含めれば容易に入手可能と考えていたからです。
実際に、自分がこの教材の実行や動画キャプチャーしたマシンは
第 4 世代　Core i3 4010 8GB メモリ　のオークション入手の ThinkPadX240 です。
WSL2 については、つい最近 Windows の標準アップデートに付加して追加が可能になったようですので、今後セットアップもさらに容易になると考えています。

アプリケーションの詳細

1.ハンズオンセミナーの概要

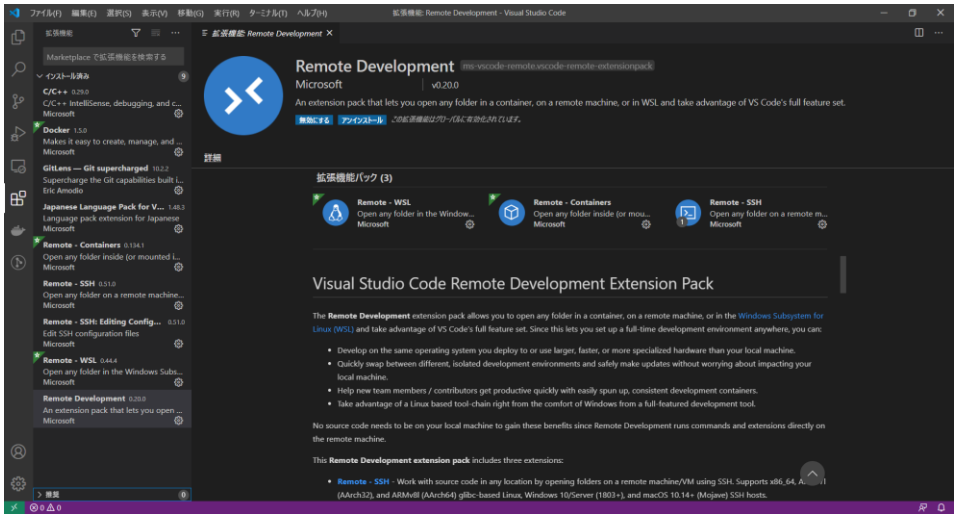
教材セミナーと重複しますが、実際にどのように実演するかをざっくりここで説明します。

1) VScode の拡張機能

実行前にまず、WSL2 および Docker for Windows のセットアップをしておきます。

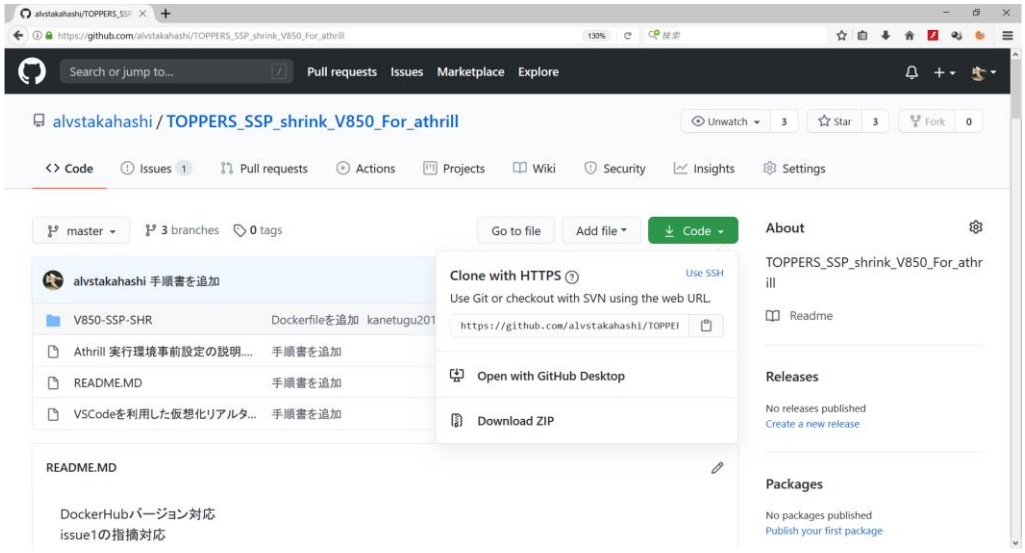
別途資料を参照の上、セットアップがまだの場合は実施します。

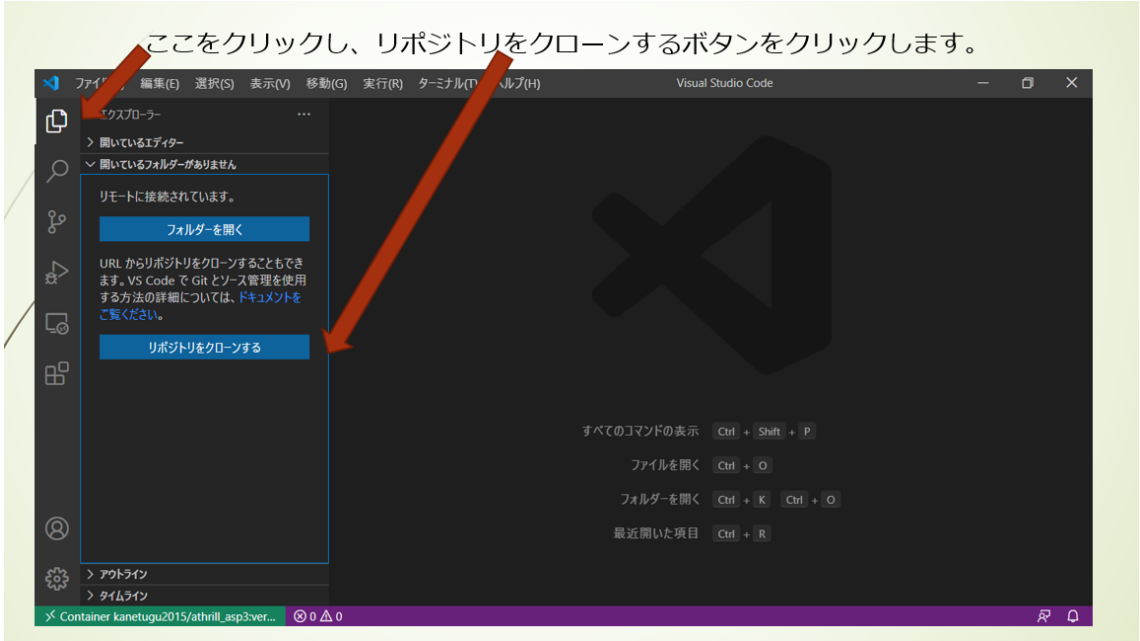
VScode では以下の画面のような拡張機能をセットアップしておきます。



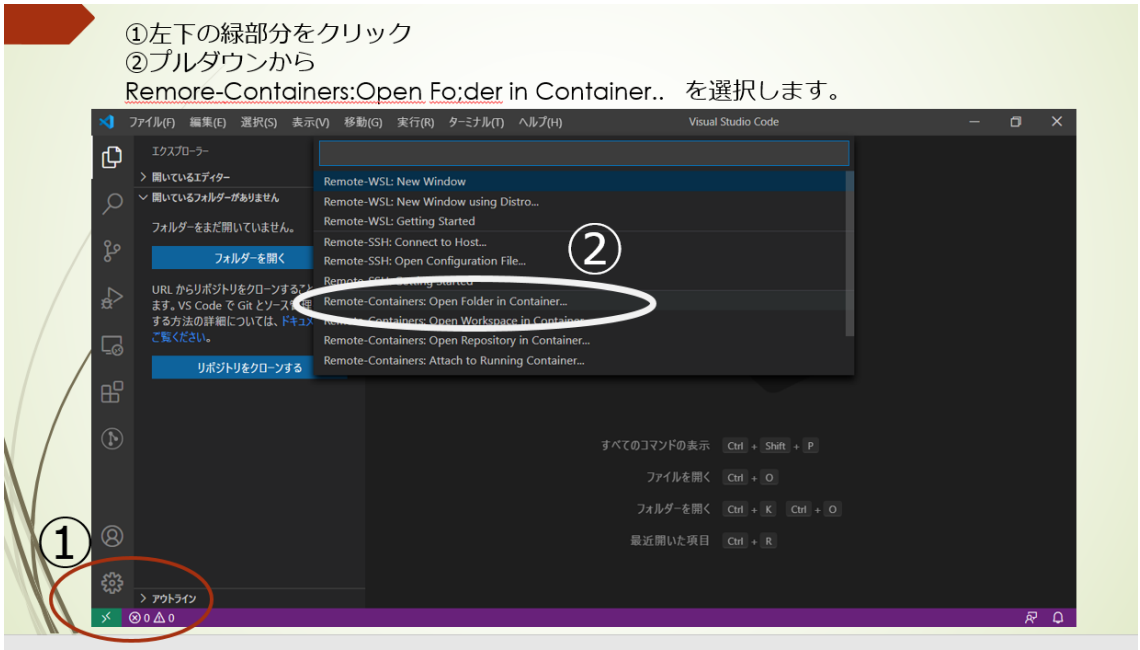
2)実施方法

①github から　TOPPERS_SSP_shrink_V850_For_athrill を PC ローカルフォルダに Clone



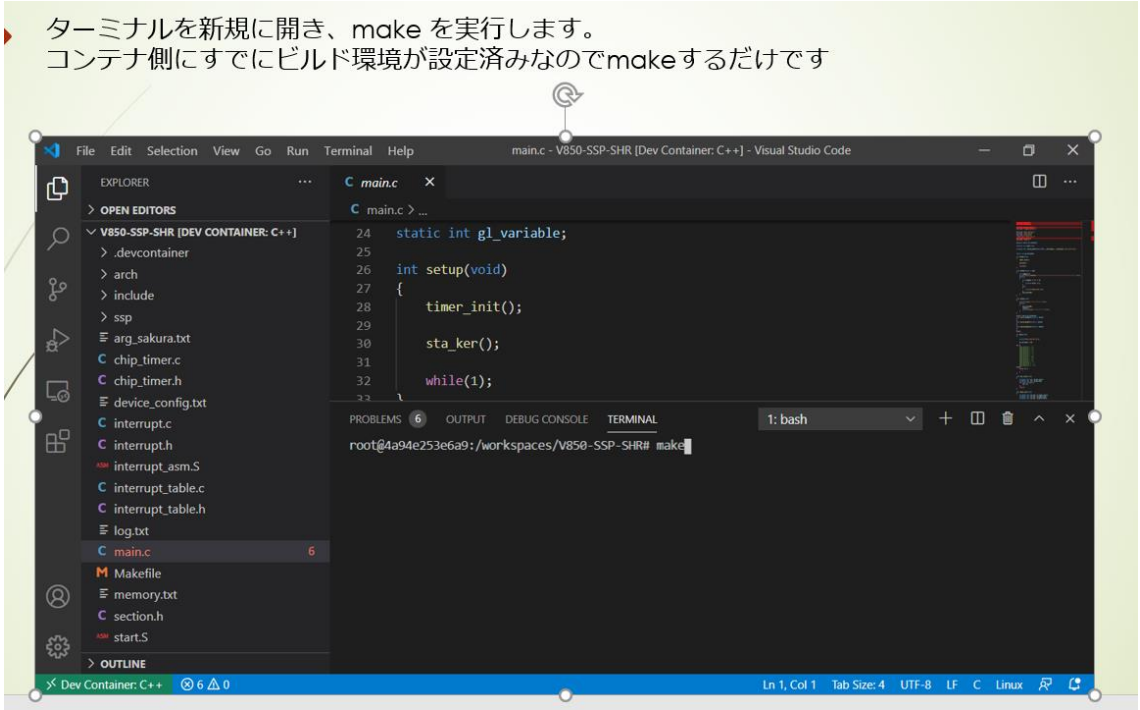


②Dockerfile の記載のあるフォルダーである Github から Clone した 1 階層下の
V850-SSP-SHR を
Remore-Containers:Open Fo;der in Container.. から指定して開きます。



ここは若干時間（数分）かかりますのでじっくり待ちます。

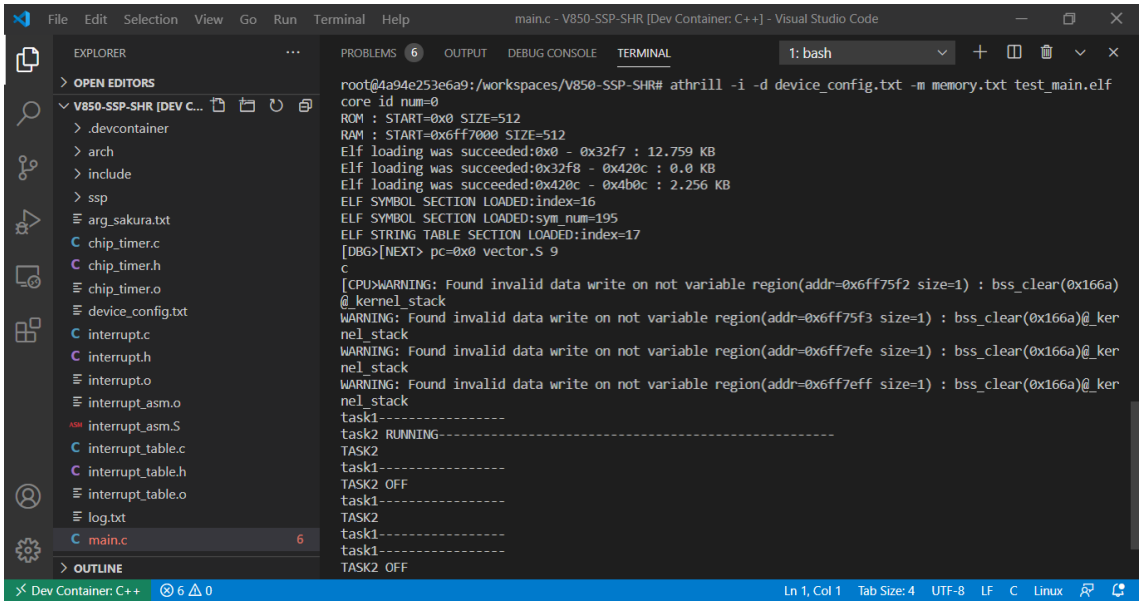
③オープンできればもう好きなように編集ビルド実行ができます。
編集は VScode でファイラーから編集、ビルド実行は新しいターミナルから可能です。



Docker に Windows ローカルフォルダがマウントされた状態になりますので、
編集は PC ローカル、ビルド、実効は Docker 環境で行う仕組みです。

Make が終わったら実行します。

athrill -i -d device_config.txt -m memory.txt test_main.elf



2.athrill 用 TOPPERS-SSP シュリンク版の今回の対応内容の詳細

GitHub に公開しているソースコード一式について今回の成果物にあたり、変更した部分について説明します。

2.1.指摘事項の修正、動作不具合の修正

DockerHUB の athrill では動作しなくなっていました。

ISSue にて、athrill 作者の森さんから指摘があがっていたので対応したところ動作するようになりました。

a)MMAP の対応

元の athrill のサンプルにあったので残していたのですが、MMAP の機能でしょうか

RAM の割り当てをする機能のように思います。　問題があるようですので削除しました。

D:¥usr¥DEV¥TOPPERSアイデア2020¥DIFF¥memory_txt.diff - Tortoise...

ファイル(F)

```
1 diff --git "a/V850-SSP-SHR/memory.txt" "b/V850-SSP-SHR/memory.txt"
2 index 9864e47..7ca412c 100644
3 --- "a/V850-SSP-SHR/memory.txt"
4 +++ "b/V850-SSP-SHR/memory.txt"
5 @@ -1,3 +1,2 @@
6  ROM, 0x00000000, 512
7  RAM, 0x06FF7000, 512
8 -MMAP, 0x7FF0000, mmap_file.bin
9
```

D:¥usr¥DEV¥TOPPERSアイデア2020¥DIFF¥ld.diff - TortoiseGitUDiff

ファイル(F)

```diff
diff --git "a/V850-SSP-SHR/v850esfk3.ld" "b/V850-SSP-SHR/v850esfk3.ld"
index dc7e579..1ffa719 100644
--- "a/V850-SSP-SHR/v850esfk3.ld"
+++ "b/V850-SSP-SHR/v850esfk3.ld"
@@ -2,7 +2,6 @@ MEMORY
 {
	INTROM(raxi) : org = 0x00000000, len = 512k
	INTRAM(wax)  : org = 0x06FF7000, len = 512k
-	INTMMAP(wax)  : org = 0x07FF0000, len = 512k
 };

@@ -57,14 +56,6 @@ SECTIONS
		. = ALIGN(4);
	} > INTRAM

-	.mmap_section	:
-	{
-		. = ALIGN(4);
-		*(.mmap_section)
-		. = ALIGN(4);
-	} > INTMMAP
-
-
	.comment 0 : { *(.comment) }
	.debug		0 : { *(.debug) }
	.line		0 : { *(.line) }
```

D:¥usr¥DEV¥TOPPERSアイデア2020¥DIFF¥main_c.diff - TortoiseGitUDiff

ファイル(F)

```diff
diff --git "a/V850-SSP-SHR/main.c" "b/V850-SSP-SHR/main.c"
index 2724287..7c27db5 100644
--- "a/V850-SSP-SHR/main.c"
+++ "b/V850-SSP-SHR/main.c"
@@ -20,7 +20,6 @@ volatile int count = 0;

unsigned char stack_data[STACK_SIZE] __attribute__ ((section(".bss_noclr")));

-unsigned char mmap_data[1024] __attribute__ ((section(".mmap_section")));

static int gl_variable;

```

b)アイドルループの処理

ready task が一つもいないときにカーネル内でアイドルループを行います。

Issue の指摘により　halt によって待つことで実行のオーバーヘッドが少なくなるとい

うことでしたので、その対応をしました。

D:¥usr¥DEV¥TOPPERSアイデア2020¥DIFF¥target_kernel.diff - TortoiseGitUDiff

ファイル(F)

```
1 diff·--git·"a/V850-SSP-SHR/arch/V850/target_kernel.h"·"b/V850-SSP-SHR/arch/V
2 index·1453f40..2e2ec23·100644
3 ---·"a/V850-SSP-SHR/arch/V850/target_kernel.h"
4 +++·"b/V850-SSP-SHR/arch/V850/target_kernel.h"
5 @@·-247,6·+247,9·@@·x_clear_int(INTNO·intno)
6 ·#define·t_lock_cpu()⟶disable_IRQ()
7 ·#define·i_lock_cpu()⟶disable_IRQ()
8 ·
9 +#define··halt_cpu()⟶⟶__asm__("halt")
10 +#define··nop_cpu()⟶⟶__asm__("nop")
11 +
12 ·/*
13 ··*··CPUロック状態の解除
14 ··*
15 @@·-286,6·+289,13·@@·extern·uint32_t⟶DEFAULT_ISTACK[];
16 ·
17 ·Inline·void·idle_loop(void)
18 ·{
19 +⟶halt_cpu();
20 +⟶nop_cpu();
21 +⟶nop_cpu();
22 +⟶nop_cpu();
23 +⟶nop_cpu();
24 +⟶nop_cpu();
25 +
26 ·⟶t_unlock_cpu();
27 ·⟶t_lock_cpu();
28 ·}
29
```

2.2.DockerFile の追加

PC のローカルファイルフォルダに規定のフォルダ名およびその下に Dockerfile を記述した場合に、Docker に接続して Windows のそのフォルダが Docker にマウントされて、いい感じになることがわかりました。

実際 Dockerfile には、DockerHub の athrill のイメージ名を記述したのみです。

ほかにこれら Docker 接続のため、上手に記述すれば　ビルドも実行もデバッグも Vscode からのメニューから行えそうですが、詳しくはよくわかっていません。今後の課題と考えています。

Dockerfile の内容

D:¥usr¥DEV¥TOPPERSアイデア2020¥DIFF¥dockerfile.diff - TortoiseGitUDiff

ファイル(F)

```
diff --git "a/V850-SSP-SHR/.devcontainer/Dockerfile" "b/V850-SSP-SHR/.devcontainer/D
index e69de29..81176d8 100644
--- "a/V850-SSP-SHR/.devcontainer/Dockerfile"
+++ "b/V850-SSP-SHR/.devcontainer/Dockerfile"
@@ -0,0 +1,6 @@
+ARG VARIANT=buster
+FROM kanetugu2015/athrill_asp3:ver3.0
+
+# [Optional] Uncomment this section to install additional packages.
+# RUN apt-get update && export DEBIAN_FRONTEND=noninteractive \
+#     && apt-get -y install --no-install-recommends <your-package-list-here>

```

今後の課題

athrill に OS を移植するという内容の 1 日程度のセミナー教材も検討したいと思っています。

TOPPERS-SSP を athrill に移植した手順を順を追って行っていくものです。

例えば以下のような内容です。

タイマー割込みだけのベアメタルプログラムに以下を追加していきます。

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部追加修正

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

以上