



μT-Kernel 3.0 BSP2 スタートガイド

e² studio & RA4M1 Clicker編

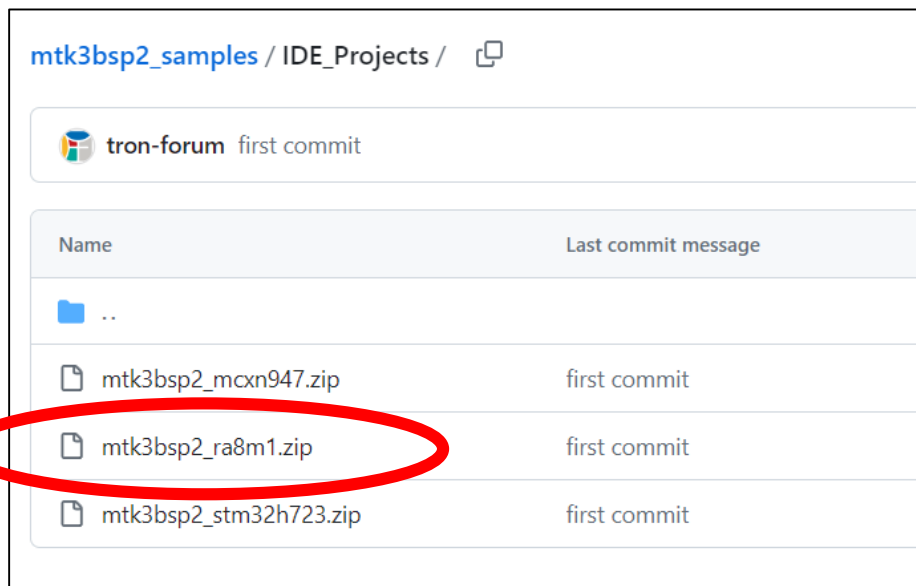


- 本スタートガイドは、 μ T-Kernel 3.0 BSP2とマイコンメーカーの提供するIDE(統合開発環境)を使用して、マイコンボードで実行するプログラムの作成、デバッグの基本的な方法を説明します。
- μ T-Kernel 3.0 BSP2やIDEなどの詳細な情報は、他のドキュメントを参照してください。

μT-Kernel 3.0 BSP2のダウンロード



- μT-Kernel 3.0 BSP2のプロジェクト mtk3bsp2_ra4m1.zipをダウンロードします。
 - https://github.com/tron-forum/mtk3bsp2_samples/tree/main/IDE_Projects
- Zipファイルを任意のディレクトリに展開します。
 - Zipファイルを展開するディレクトリのパス名に日本語が入らないように注意してください。



mtk3bsp2_samples / IDE_Projects /

tron-forum first commit

Name	Last commit message
..	
mtk3bsp2_mcxn947.zip	first commit
mtk3bsp2_ra8m1.zip	first commit
mtk3bsp2_stm32h723.zip	first commit



- Flexible Software Package (FSP)のインストーラを以下よりダウンロードしインストールします。

- <https://www.renesas.com/jp/ja/software-tool/flexible-software-package-fsp>
- 使用するFSPのバージョンは**v5.4.0**です
- 統合開発環境 (e² studio) も一緒にインストールされます。
- e² studio、FSPについて詳細は上記のWebサイトをご覧ください。

RENESAS

設計リソース > Software - Middleware, Drivers, OS > RA Flexible Software Package (FSP)

RA Flexible Software Package (FSP)

Software Package

マニュアルSWをダウンロード

Renesas Flexible Software Package (FSP) v5.4.0 User's Manual

Renesas Flexible Software Package (FSP) User's Manual (Web Version - HTML)

概要 ダウンロード ドキュメント 設計・開発 サポート ビデオ&トレーニング 追加詳細

概要

説明 特長 リリース情報 ターゲットデバイス

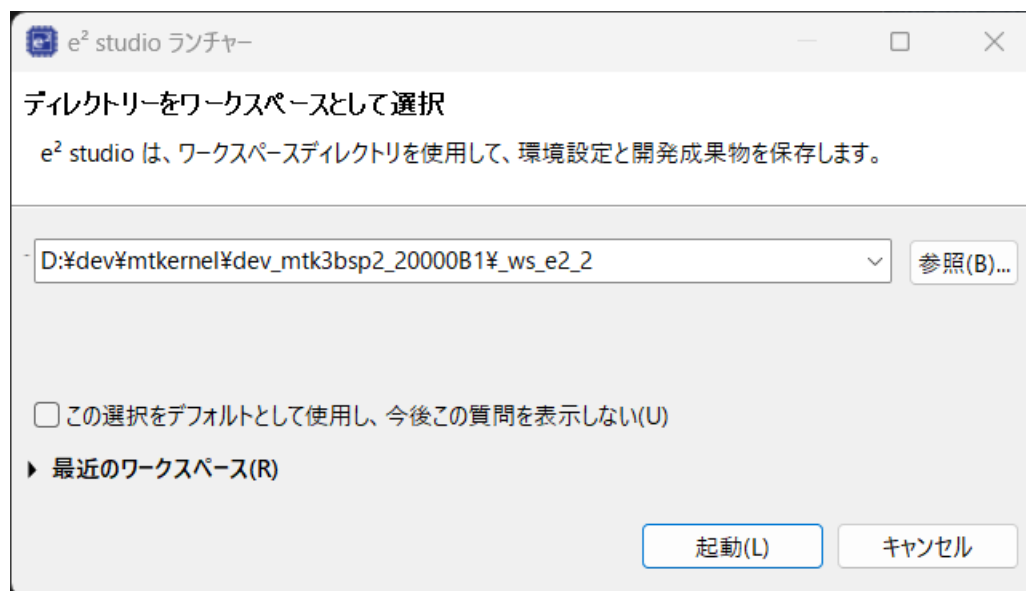
最新FSP (v5.4.0) をダウンロード:

FSPプラットフォームインストーラ (e² studio IDE、ツールチェーン、FSPパックを含む):

- FSP Windows Platform Installer
- FSP Linux Platform Installer
- FSP macOS Platform Installer



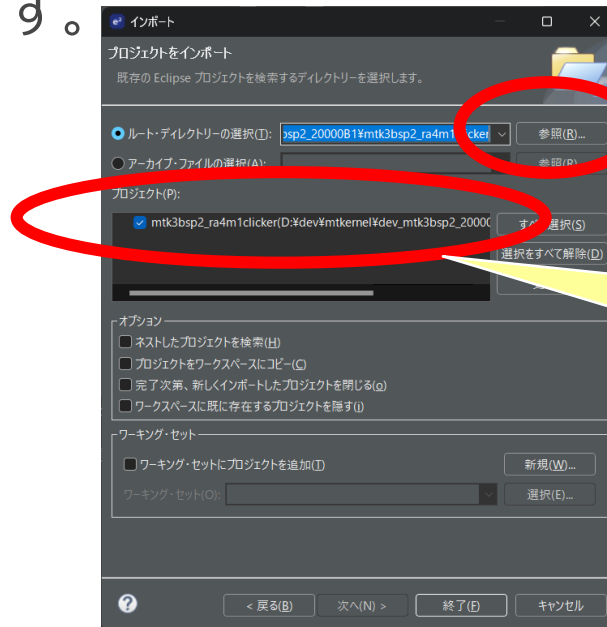
- インストールしたe² studioを実行します。
 - 起動時にワークスペースを聞かれます。任意のディレクトリを指定してください。ここにIDEの各種情報が保存されます。



プロジェクトのインポート



- ① メニュー[ファイル]→[インポート]を選択します。
- ② 開いたダイアログから[一般]→[既存プロジェクトをワークスペースへ]を選択し[次へ]を押下します。
- ③ [ルート・ディレクトリの選択]の[参照]ボタンを押し、BSP2のプロジェクトのディレクトリを指定します。
- ④ BSP2のプロジェクトが表示されていることを確認のうえ[終了]を押下します。



③ [ルート・ディレクトリの選択]の[参照]ボタンを押し、BSP2のプロジェクトのディレクトリを指定します。

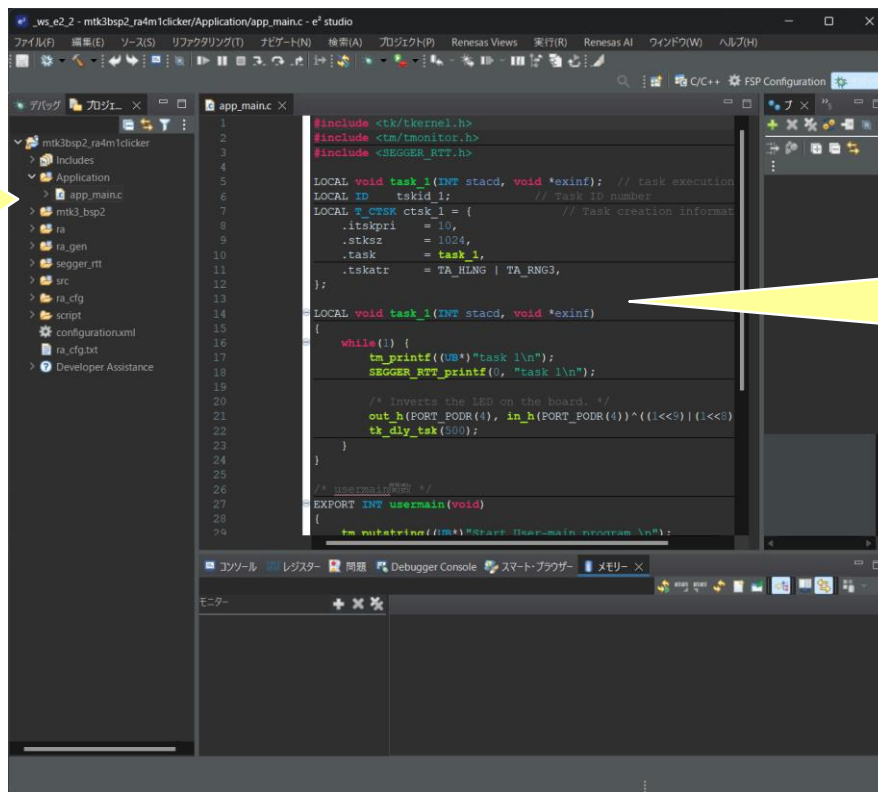
④ BSP2のプロジェクトが表示されていることを確認のうえ[終了]を押下します。

プロジェクトの表示



- インポートが正常に終了すると、プロジェクトマネージャーにμT-Kernel 3.0 BSP2のプロジェクトが表示されます。
- 表示されているファイルをダブルクリックすると、その内容が表示され、編集ができます。

BSP2のプロジェクトが表示されます。



BSP2のプロジェクトの選択したファイルが表示され、編集ができます。

プロジェクトのビルド



- プロジェクトマネージャーのプロジェクト名を右クリックし、[プロジェクトのビルド]を選択します。
- プロジェクトのビルドが開始され、正常に終了すると「Build Finished.」が表示されます。

プロジェクト名を
右クリックし、[プ
ロジェクトのビル
ド]を選択します。

```
#include <tk/tkernel.h>
#include <tm/tmonitor.h>
#include <SEGGER_RTT.h>

LOCAL void task_1(INT stacd, void *exinf); // task execution
LOCAL ID tskid_1; // Task ID number
LOCAL T_CTSK ctsk_1 = {
    .tskpri = 10,
    .tsksz = 1024,
    .task = task_1,
    .tskatr = TA_HLNG | TA_RNG3,
};

LOCAL void task_1(INT stacd, void *exinf)
{
    while(1) {
        tm_printf((UB*)"task 1\n");
        SEGGER_RTT_printf(0, "task 1\n");

        /* Inverts the LED on the board. */
        out_h(PORT_PODR(4), in_h(PORT_PODR(4))^(1<<9)|(1<<8));
        tk_dly_task(500);
    }
}

/* main関数 */
EXPORT INT usermain(void)
{
    tm_printf((UB*)"Start Usermain program\n");
}
```

CDIビルド: コンソール [mtk3bsp2_ra4m1clicker]
Building target: mtk3bsp2_ra4m1clicker.elf
arm-none-eabi-objcopy -O srec "mtk3bsp2_ra4m1clicker.elf" "mtk3bsp2_ra4m1clicker.srec"
arm-none-eabi-size -format=berkeley "mtk3bsp2_ra4m1clicker.elf"
text data bss dec hex filename
34616 824 12104 47544 b9b6 mtk3bsp2_ra4m1clicker.elf
16:05:46 Build Finished. 0 errors, 0 warnings. (took 4s.11)

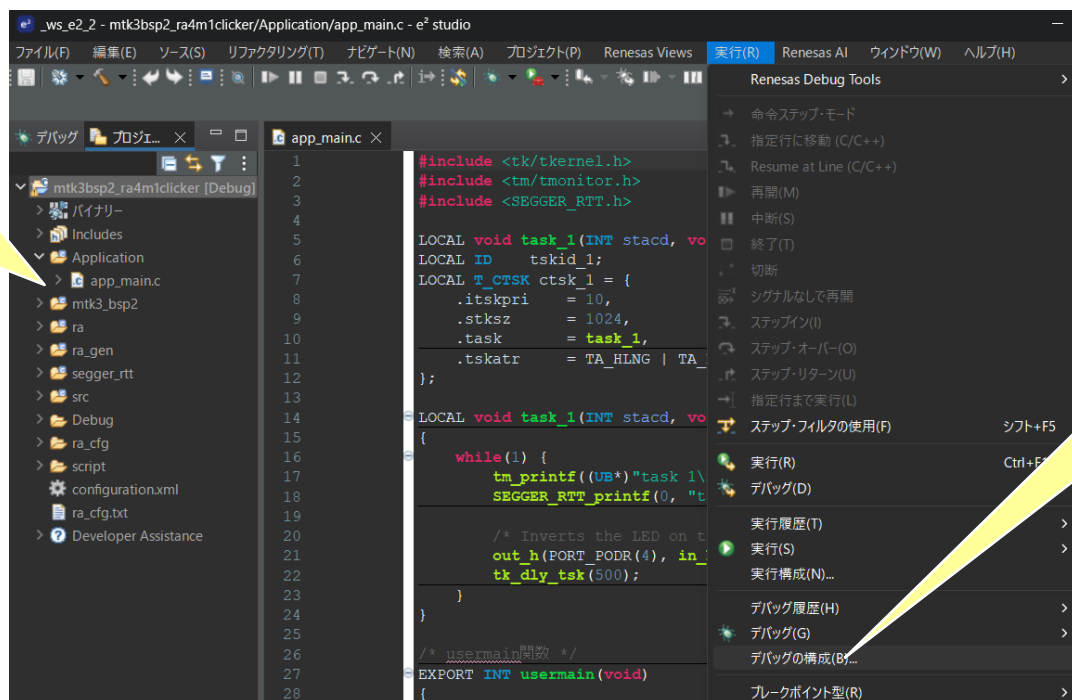
ビルドが正常に終
了すると「Build Fi
nished.」が表示さ
れます。

プログラムの実行とデバッグ(1)



- ボード(RA4M1 Clicker)とPCをUSBで接続します。
- プロジェクトマネージャーのプロジェクト名を選択した状態でメニュー[実行]から[デバッグの構成]を選びます。

プロジェクト名
を選択します。



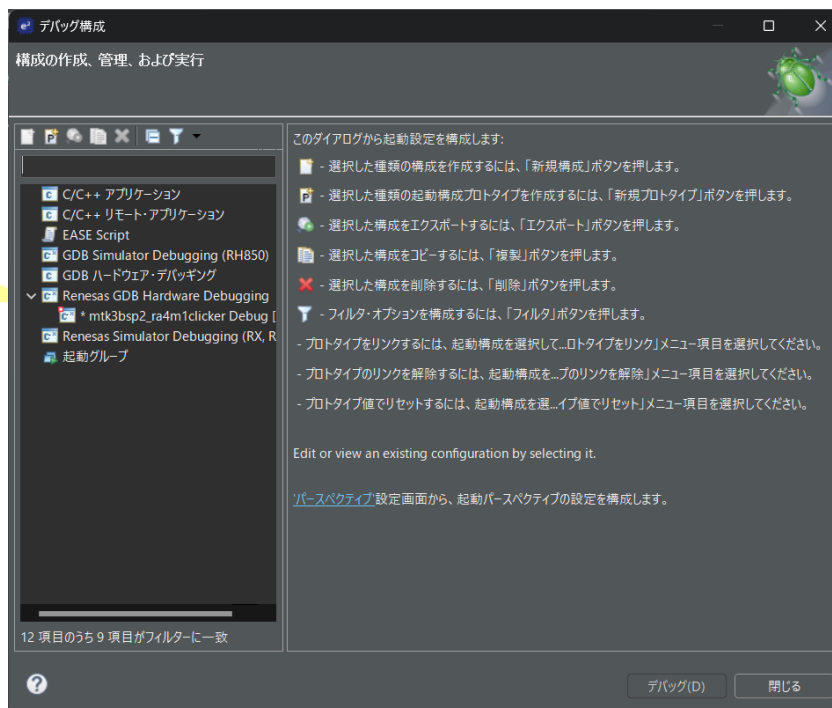
メニュー[実行]から
[デバッグの構成]
を選びます。

プログラムの実行とデバッグ(2)



- 表示されたダイアログから[Renesas GDB Hardware Debugging]の[mtk3bsp2_ra4m1clicker Debug]を選択します。
 - [mtk3bsp2_ra4m1clicker Debug]が表示されていない場合は、[Renesas GDB Hardware Debugging]をダブルクリックしてください。直前にビルドしたプロジェクトが対象となります。ビルドの直後に操作してください。

[mtk3bsp2_ra4m1clicker Debug]を選択します。

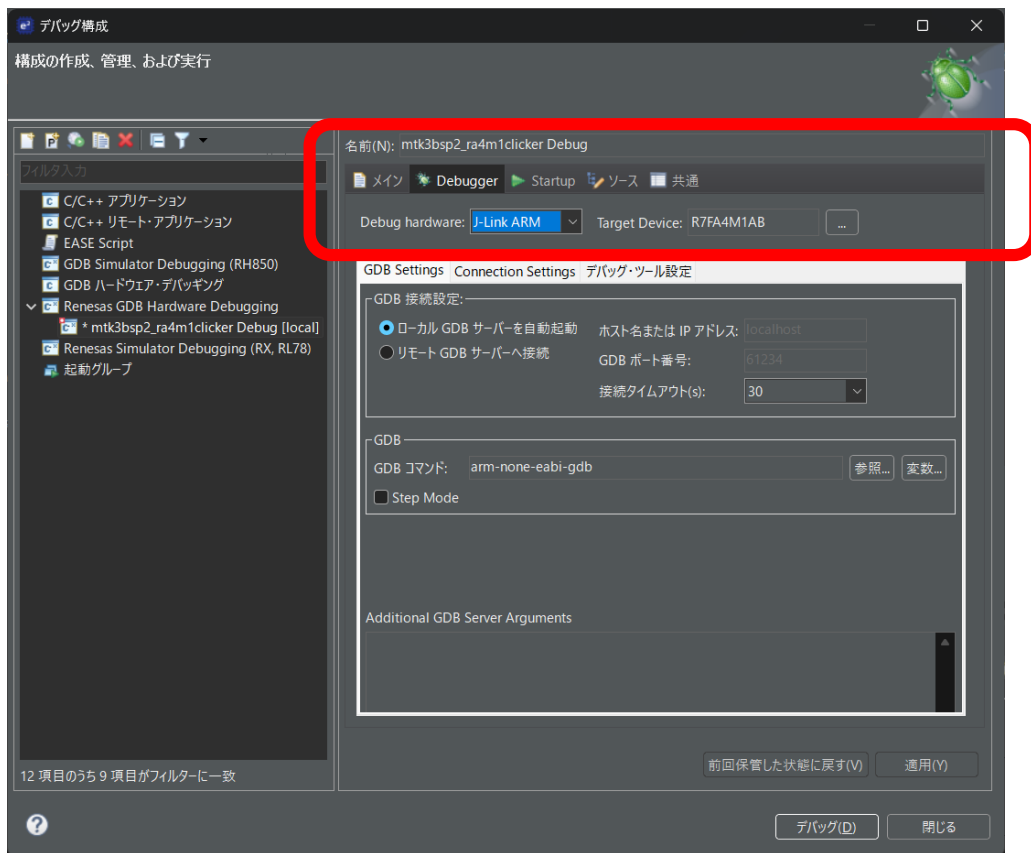


プログラムの実行とデバッグ(3)



■ [Debugger]タブを選択し以下を確認します。

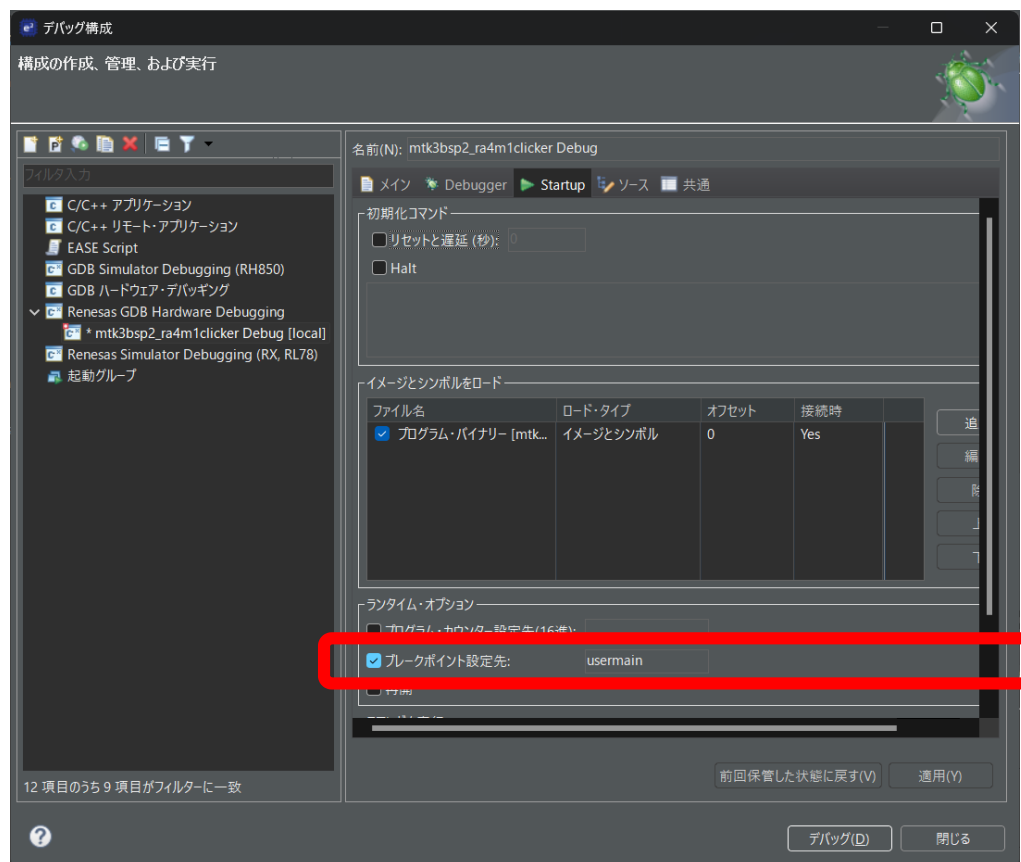
- Debug hardware : J-Link ARM
- Target Device : R7FA4M1AB



プログラムの実行とデバッグ(4)



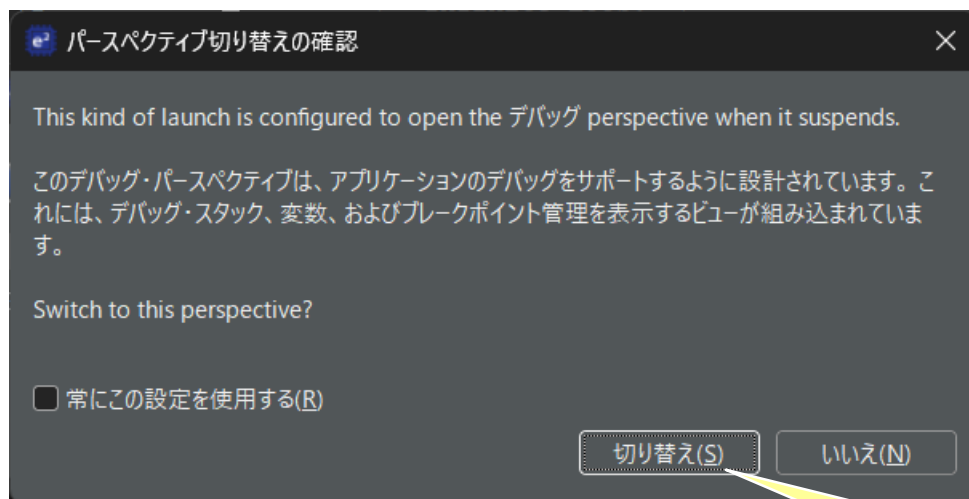
- [Startup]タブを選択し以下を設定します。
 - ブレークポイント設定先: usermain
 - ▶ デバッグ実行時にブレークポイントを設定する関数名です



プログラムの実行とデバッグ(5)



- ダイアログの[デバッグ]ボタンを押すと、実行プログラムがボードに転送されて実行されデバッグが始まります。
- [デバッグ・パースペクティブ]への切り替えが表示されますので[切り替え]ボタンを押下します。デバッグ画面に切り替わります。

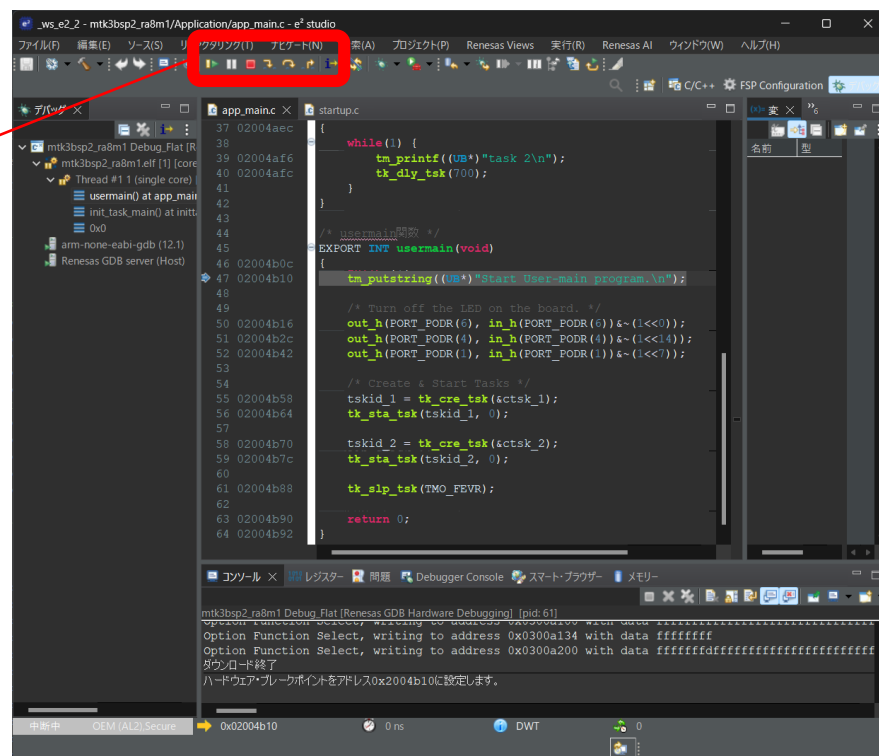
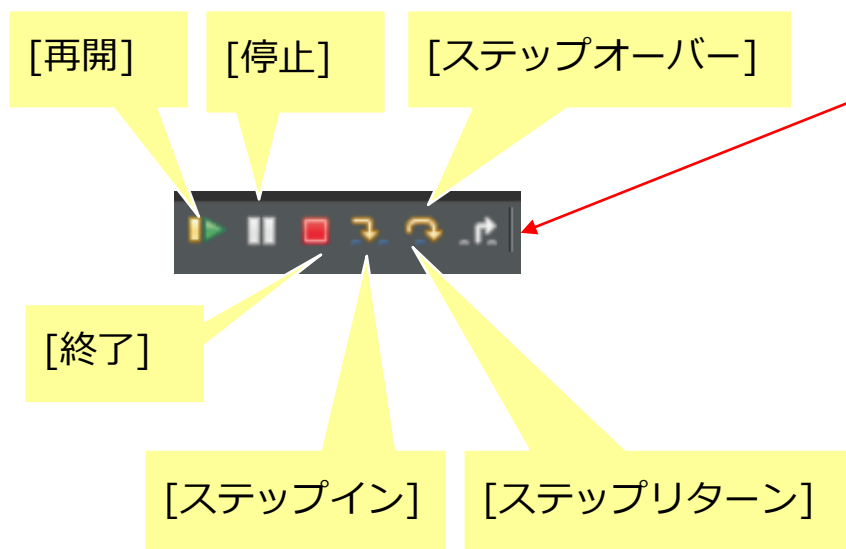


[切り替え]を押下します。

プログラムの実行とデバッグ(6)



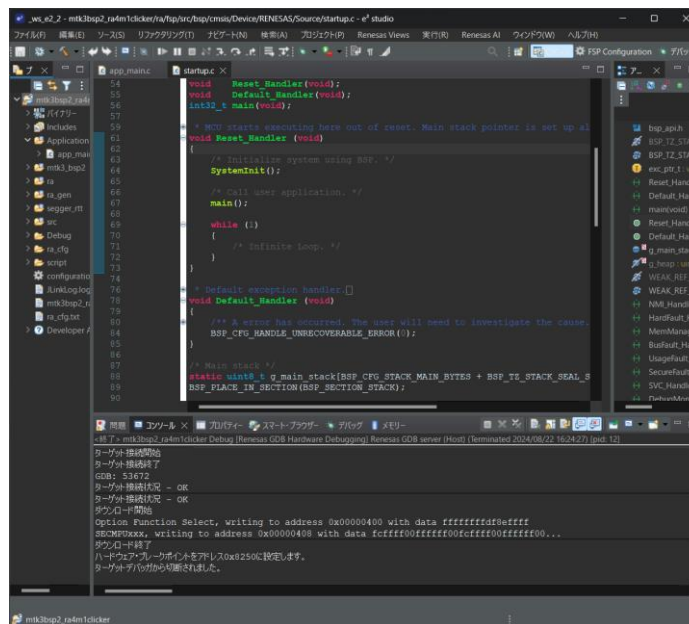
- 最初にリセットハンドラ(Reset_Handler)で停止します
- 実行を開始するとapp_main.cのusermain関数でブレークします。
- メニューバーのボタンから基本的なデバッグ操作が可能です。



ユーザプログラムの作成



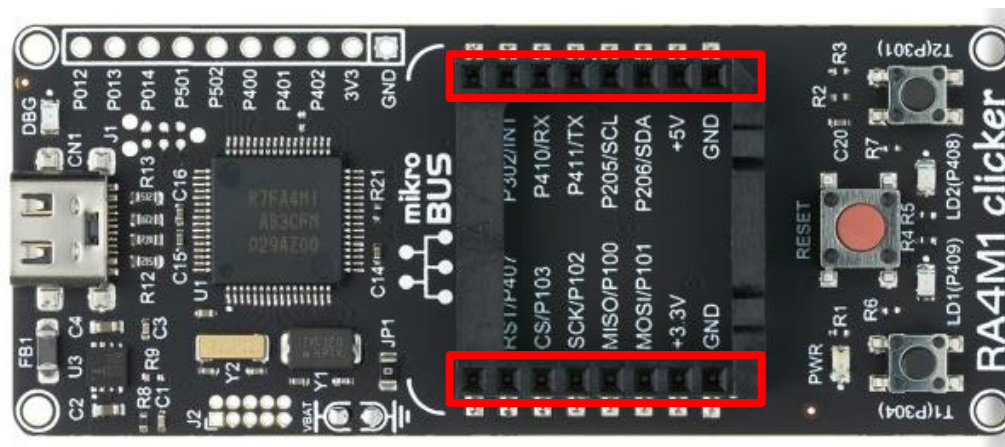
- μ T-Kernel 3.0 BSP2のApplicationディレクトリにユーザプログラムを記述します。
 - ユーザプログラムのディレクトリは任意の場所に作成可能です。
 - 他のディレクトリから独立に作成しておくで、BSP2のバージョンアップの際に移行が楽になります。
- 初期状態では、タスクを2つ実行し、それぞれのタスクがボード上のLEDの点滅とデバッグ用シリアル出力を行うプログラムがapp_main.cファイルに記述されています。





- μ T-Kernel 3.0 BSP2は、A/DコンバータとI²C通信のサンプルデバイスドライバが組み込まれています
 - サンプルデバイスドライバからはRA4M1 ClickerボードのmikroBUSコネクタの以下の信号が使用可能です
 - 他の信号もプロジェクトのコンフィギュレーション等の変更により使用できます

信号名	デバイス名	機能
AN	hadca	アナログ信号入力
SCL/SDA	hiica	I ² C通信

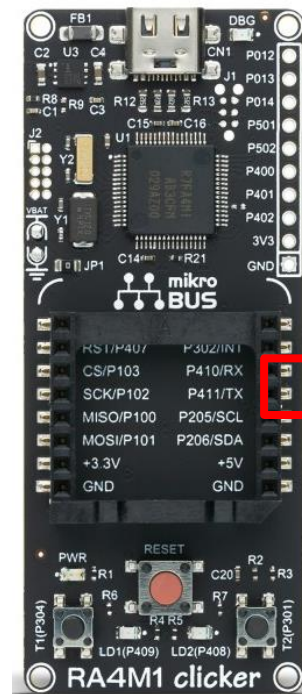


mikroBUSコネクタ

デバッグ用シリアル通信出力



- プログラムからのtm_printf関数によりデバッグ用シリアル通信に出力できます。
 - RA4M1 ClickerボードのmikroBUSのシリアル信号に出力されます。
- PCでターミナルソフトを実行すると、デバッグ用シリアル出力を表示することができます。
 - PCのターミナルソフトにはTera Termなどが使用できます。
 - シリアル通信の設定は以下にしてください。



SEGGER RTTによるデバッグ用出力



- J-LinkデバッガにはRTTというデバッグ用の通信機能があります。
 - SEGGER_RTT_printf関数によりデバッグ用通信に出力できます。
 - 詳細は以下をご覧ください
<https://www.segger.com/products/debug-probes/j-link/technology/about-real-time-transfer/>
- PCでJ-Link RTT Viewerを実行します。
 - 起動時に以下に設定します。

