μT-Kernel3.0 BSP マニュアル

- RX65N Renesas Target Board編 -

Version.01.00.00

2022.4.28

更新履歴

版数(日付)	内　容
1.00.00 (2022.04.28)	● 初版

目次

1. 概要

本書は、μT-Kernel 3.0 BSP (Board Support Package)の使用方法を説明する。
μT-Kernel 3.0 BSP は、特定のマイコンボード等のハードウェアに対して移植したμT-Kernel 3.0 の開発および実行環境一式を提供するものである。

1.1 対象とするハードウェアと OS

開発対象のハードウェアおよび OS は以下である。

分類	名称	備考
マイコン	RX65N	ルネサス エレクトロニクス
OS	μT-Kernel3.0	TRON フォーラム ※1
実機 (マイコンボード)	Target Board for RX65N	ルネサス エレクトロニクス

※1 μT-Kernel 3.0 の正式版は以下の GitHub のリポジトリから公開

https://github.com/tron-forum/mtkernel_3

1.2 対象とする開発環境

対象とする開発環境は以下である。開発を行うホスト PC の OS は Windows とする。確認は Windows 10 にて行った。

分類	名称	備考
開発環境	e^{2}sutdio Version: 2022-01 (22.1.0)	ルネサス エレクトロニクス

※ バージョン番号は確認した中で最新のバージョンを示している。

1.3 デバイスドライバ

μT-Kernel 3.0 BSP では、TRON フォーラムが提供するμT-Kernel 3.0 のサンプル・デバイスドライバを、対象実機用に移植して実装している。
以下に Target Board for RX65N の BSP のデバイスドライバを示す。

種別	対象 I/O デバイス	デバイス名
シリアル通信デバイスドライバ	SCI0	sera
	SCI1	serb
	SCI2	serc
	SCI3	serd

	SCI4	sere
	SCI5	serf
	SCI6	serg
	SCI7	serh
	SCI8	seri
	SCI9	serj
	SCI10	serk
	SCI11	serl
	SCI12	serm
A/D 変換デバイスドライバ	S12AD	adca
	S12AD1	adcb
I^2C 通信デバイスドライバ	RIIC0	iica
	RIIC1	iicb
	RIIC2	iicc

1.4 関連ドキュメント

分類	名称	発行
OS 仕様	μT-Kernel 3.0 仕様書 (Ver.3.00.00)	TRON フォーラム ※1 TEF020-S004-3.00.00
デバイス ドライバ	μT-Kernel 3.0 デバイスドライバ 説明書(Ver.1.00.2)	TRON フォーラム ※2 TEF033-W007-210331

※1 TRON フォーラム Web ページから公開

https://github.com/tron-forum/mtkernel_3

※2 μT-Kernel 3.0 正式版に含まれる（以下の GitHub のリポジトリから公開）

https://github.com/tron-forum/mtkernel_3

2. 開発環境の準備

μT-Kernel 3.0 BSP を使用するにあたり、以下の手順で開発環境の準備を行う。

2.1 e²studio のインストール

以下から使用する PC の OS に合わせて、e²studio をダウンロードする。

e²studio の Web ページ

https://www.renesas.com/jp/ja/software-tool/e-studio

インストーラがダウンロードされるので、それを実行し、指示に従ってインストールを進める。

3. プロジェクトの作成

e²studio にμT-Kernel 3.0 BSP のプロジェクトを以下の手順で作成する。

3.1 GitHub からのプロジェクトの入手

TRON フォーラムの GitHub の以下のレポジトリから BSP のプロジェクトが公開されている。

https://github.com/tron-forum/mtk3_bsp

対象ハードウェア毎にブランチが作成されている。Target Board for RX65N の BSP のブランチ名は「rx65n_rtb」である。また最新のリリース版は、GitHub の Release から取得することができる。

3.2 プロジェクトのファイル構成

プロジェクトのディレクトリおよびファイルの構成はμT-Kernel 3.0 の正式リリース版に準じ以下のように構成される。

ディレクトリまたはファイル名	内容
app_program	アプリケーション・プログラム用のディレクトリ
build_make	Make 構築ディレクトリ（BSP では使用しない）

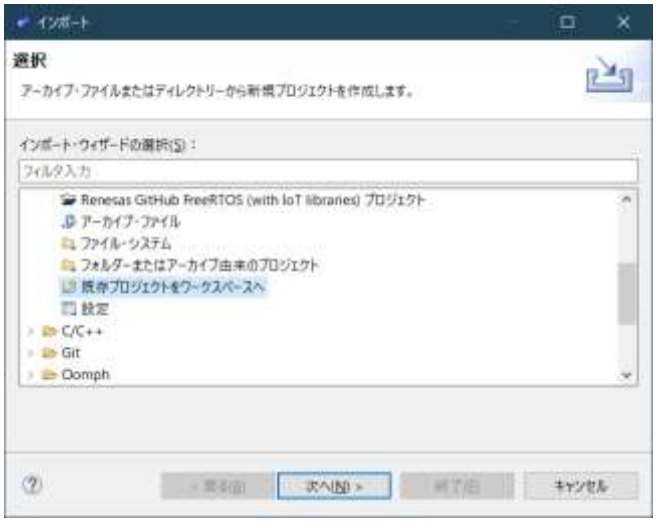
config	コンフィギュレーション
device	デバイスドライバ
docs	ドキュメント
etc	リンカファイル等
include	各種定義ファイル
kernel	μT-Kernel 本体
lib	ライブラリ
.settings	e2studio の設定ファイル
.cproject	e2studio のプロジェクトファイル
その他ファイル	e2studio の各種ファイルなど

ディレクトリ「app_program」に作成するアプリケーション・プログラムを格納する。初期状態では、サンプルコードを記述した app_main.c ファイルが格納されている。
ユーザが通常操作するのは、このディレクトリ「app_program」である。

3.3 プロジェクトのインポート

前項で入手したプロジェクトを e2studio の開発環境に以下の手順で取り込む。

(1) 「ファイル」メニューから「インポート」を選択する。
(2) インポートのダイアログから「一般」→「既存プロジェクトをワークスペースへ」を選択する。



(3) 「ルートディレクトリの選択」を押し、前項で入手したプロジェクトのディレクトリを指定する。
(4) 「プロジェクト」に「rtb_rx65n」が表示されるので、それをチェックする。

(5) 「終了」ボタンを押下すると、プロジェクトがワークスペースに取り込まれる。

プロジェクトのインポートが完了すると、e2studio のプロジェクト・エクスプローラに「rtb_rx65n」が表示される。

3.4 プロジェクトのビルドの確認

プロジェクトのμT-Kernel 3.0 のプログラムがビルド出来ることを確認する。
プロジェクト「rtb_rx65n」が選択されている状態でメニュー「プロジェクト」→「プロジェクトのビルド」を選択する。
プログラムのビルドが実行され、「コンソール」に「Build Finished. 0 errors」が表示されれば、正常にビルドが完了している。

4. アプリケーション・プログラムの作成

4.1 アプリケーション・プログラムのソースファイル

本プロジェクトでは、アプリケーション・プログラムのソースファイルは、プロジェクトのディレクトリ下の「app_program」ディレクトリに置くことを前提としている。
初期状態では、サンプルコードを記述した app_main.c ファイルが置かれている。ユーザは本ファイルの内容を変更することにより、プログラムを記述することができる。また、複数のファイルが存在する場合は、このディレクトリ下に置くことができる。

4.2 ワーキング・セットの選択

e2studio ではプログラムの各種設定をワーキング・セットとしてプロジェクトに対して作成する。ワーキング・セットは、一つのプロジェクトに複数作ることができる。
本プロジェクトは初期状態では、「Release」と「HardwareDebug」の二つのワーキング・セットが作られている。通常、デバッグ時は「HardwareDebug」を使用する。
ワーキング・セットは、メニュー「Project」→「Build configurations」→「Set Active」から選択することができる。

「Release」と「HardwareDebug」の設定の違いを以下に示す。なお、ワーキング・セットの各種設定は、アプリケーション・プログラムの必要に応じて設定・変更を行う。

項目	HardwareDebug	Release
Optimaization Level	None(-O0)	Optimize more(-O2)
Debug Level	Default(-g2)	Minimal(-g1)

4.3 実行プログラムのビルド

プロジェクトが選択されている状態で、メニュー「プロジェクト」→「プロジェクトのビルド」を選択すると、プログラムがビルド（コンパイル、リンク）され、実行コード・ファイル「mtk3bsp_rtb_rx65n.elf」が生成される。
実行コード・ファイルは、プロジェクトのディレクトリ下のワーキング・セットと同名のディレクトリに生成される。

5. 実機でのプログラム実行

ビルドし生成した実行コードを実機上で実行する方法を説明する。

5.1 実行環境の準備

e2studio を実行しているパソコンと実機ボード（Target board for RX65N）を USB ケーブルで接続する。パソコンから実機へのプログラムの転送およびデバッグはこの USB を経由して行うことができる。

Target board for RX65N は USB シリアル通信の機能を持たないため、デバッグ用シリアル出力は使用できない。

5.2 プログラムのデバッグ実行

プログラムのデバッグ実行は、以下の手順で行う。なお(1)から(3)までの操作は最初に 1 回のみ行うだけでよい。

(1) e2studio のメニューからメニュー「実行」→「デバッグの構成」を選択し、開いたダイアログから項目「Renesas GDB Hardware Debugging」を選択する。

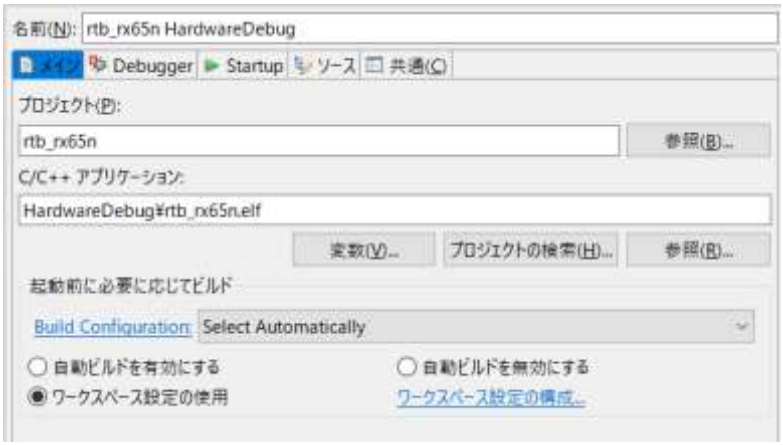
(2) 「Renesas GDB Hardware Debugging」下の「rtb_rx65n HardwareDebug」を選択する。もし「rtb_rx65n HardwareDebug」が無い場合は、「新規の起動構成」ボタンを押し構成を追加する。

(3) 作成した構成を選択し以下の設定を行う。その他の設定についても必要に応じて変更すること。

- 「メイン」タブ

  プロジェクト：前項で作成したプロジェクトを指定

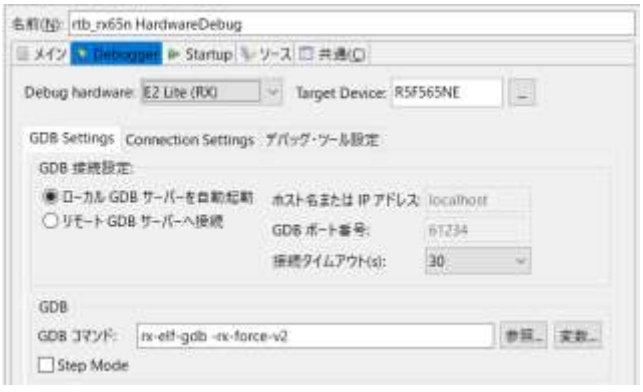
  C/C++ アプリケーション：ビルドした ELF ファイル



- 「Debugger」タブ

  Debug haedware：「E2 Lite(RX)」を選択

  Target Device：「R5F565NE」を選択



「Connection Setting」で以下を設定

・メイン・クロック・ソース：HOCO

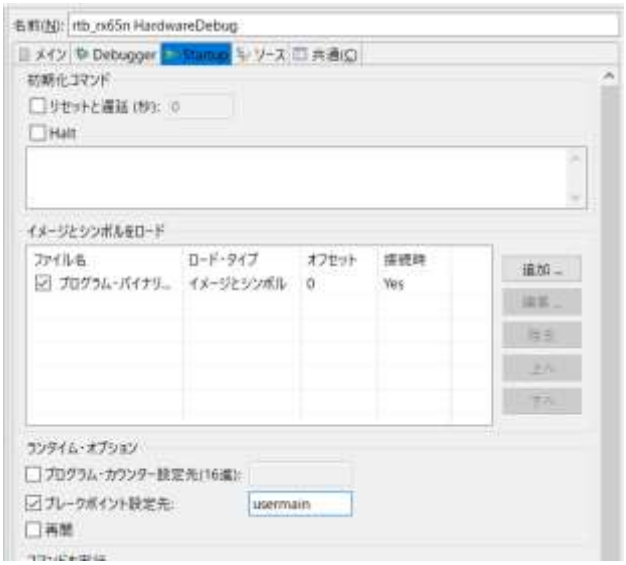
・動作周波数：120

・接続タイプ：Fine

・エミュレータから電源を供給する：「いいえ」

- 「Startup」タブ

  Set breakpoint at：　「usermain」を入力

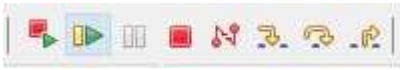


(4) デバッグ開始

「デバッグ」ボタンを押すとプログラムが実機に転送され、e²studio はデバッグ画面に変わる。プログラムは前項において設定したブレークポイント(usermain 関数の先頭)で停止する。

(5) プログラム実行

ブレークにより一時停止しているプログラムは再開ボタンの押下で実行を再開する。



サンプル・プログラムは、実機ボード（Tartget Board for RX65N）上の LED(LED0～LED1)を点滅させる。

上記の操作でデバッグ構成が作成されたので、2 回目以降のデバッグは、既に作成済みのデバッグ構成を使用することができる。

5.3 プログラムの実行

前項のデバッグ実行を行うことにより、実行コードはマイコンの Flash ROM に書き込まれる。よって、e²studio を使用せずに単独でボードに電源を入れれば、プログラムは実行される。

以上