# μT-Kernel3.0 BSP マニュアル

## - STM32L476 Nucleo-64 編 -

Version.01.00.01

2022.04.14

更新履歴

| 版数(日付) | 内　容 |
|---|---|
| 1.00.01<br>(2022.02.14) | 表記統一<br>「STM32 Nucleo-64」→「STM32L476 Nucleo」など |
| 1.00.00<br>(2021.11.15) | ● 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 | STM32L476RG<br>(ARM Cortex-M4 コア) | ST マイクロエレクトロニクス製 |
| OS | μT-Kernel3.00.05 | TRON フォーラム ※1 |
| 実機<br>(マイコンボード) | STM32L476 Nucleo-64 | ST マイクロエレクトロニクス製 |

※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 | STM32CubeIDE v1.8.0 | STMicroelectronics |

※ 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 STM32L476 Nucleo-64 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 | USART1 | sera |
| | USART2 | serb |
| | USART3 | serc |
| A/D 変換デバイスドライバ | ADC1 | adca |

| | | |
|---|---|---|
| | ADC2 | adcb |
| | ADC3 | adcc |
| I²C 通信デバイスドライバ | I2C1 | iica |
| | I2C2 | iicb |
| | I2C3 | 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<br>(Ver.3.00.00) | TRON フォーラム ※1<br>TEF020-S004-3.00.00 |
| 実装仕様書 | STM32L4 IoT-Engine 向け<br>実装仕様書 (Ver.01.00.02) | TRON フォーラム ※2<br>TEF033-W008-211115 |
| デバイス<br>ドライバ | μT-Kernel 3.0 デバイスドライバ<br>説明書(Ver.1.00.2) | TRON フォーラム ※2<br>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 STM32CudeIDE のインストール

以下から使用する PC の OS に合わせて、STM32CudeIDE をダウンロードする。

**STM32CudeIDE のダウンロードページ**

https://www.st.com/ja/development-tools/stm32cubeide.html

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

STM32CudeIDE 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。STM32L476 Nucleo-64 の BSP のブランチ名は「stm32l476_nucleo」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） |

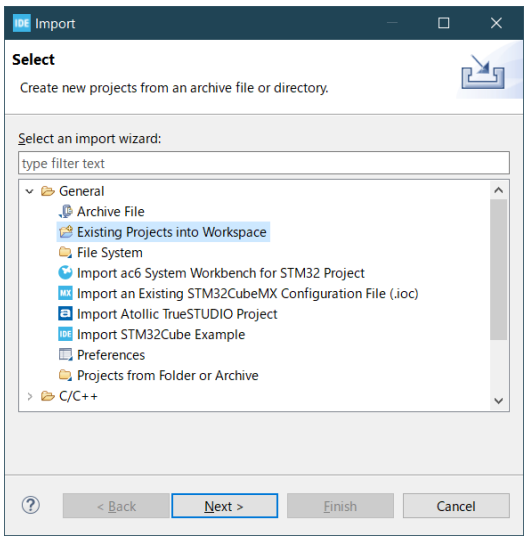
| | |
|---|---|
| config | コンフィギュレーション |
| device | デバイスドライバ |
| docs | ドキュメント |
| etc | リンカファイル等 |
| include | 各種定義ファイル |
| kernel | μT-Kernel 本体 |
| lib | ライブラリ |
| .settings | STM32CubeIDE の設定ファイル |
| .cproject | STM32CubeIDE のプロジェクトファイル |
| その他ファイル | STM32CudeIDE の各種ファイルなど |

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。

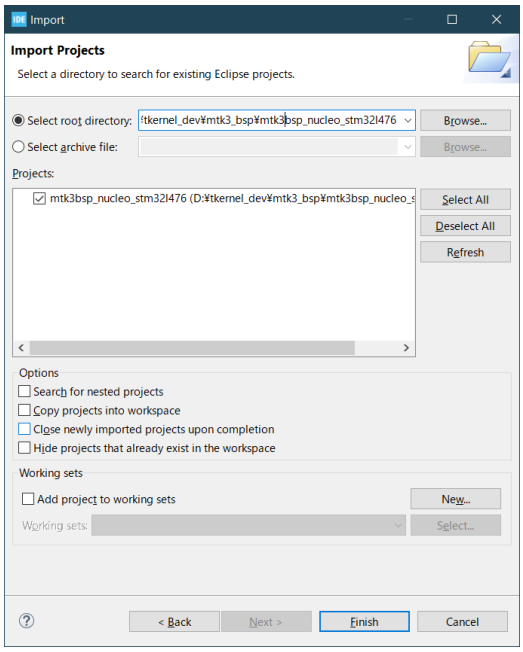
## 3.3 プロジェクトのインポート

前項で入手したプロジェクトを STM32CudeIDE の開発環境に以下の手順で取り込む。

(1) メニュー「Import」を選択する。
(2) インポートのダイアログから「General」→「Existing Projects into Workspace」を選択する。



(3) 「Select root directory」で「Browse」ボタンを押し、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) 「Projects」に「mtk3bsp_nucleo_stm32l476」が表示されるので、それをチェックする。




(5) 「Finish」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、STM32CudeIDE のプロジェクト・エクスプローラに「mtk3bsp_nucleo_stm32l476」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「mtk3bsp_nucleo_stm32l476」が選択されている状態で、メニュー「Project」→「Build project」を選択する。
プログラムのビルドが実行され、「Console」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

STM32CudeIDE 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「Debug」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「Debug」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「Debug」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 | Debug | Release |
|---|---|---|
| 最適化レベル | なし(-O0) | さらに最適化(-O2) |
| デバッグ・レベル | 最大(-g3) | 最小(-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_nucleo_stm32l476.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

STM32CudeIDE を実行しているパソコンと実機ボード（STM32L4723 Nucleo-64）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。

また、パソコンから仮想 COM ポート(シリアル通信ポート)して、実機ボードとの接続が認識される。パソコン上でターミナルエミュレータ（例：TeraTerm ※1）を実行することにより、実機ボードと通信を行うことができる。

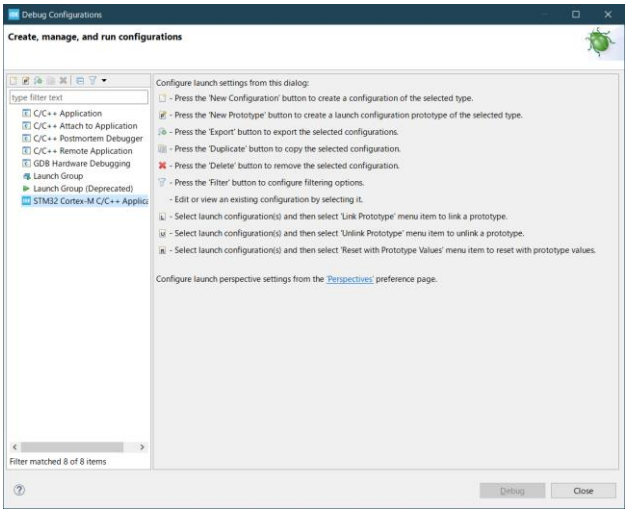
μT-Kernel 3.0 BSP では、デバッグ用シリアル出力をこの仮想 COM ポートに送信する。また、μT-Kernel 3.0 のシリアル通信デバイスを使用してパソコンと通信を行うことが可能である。

※1 Tera Term Home Page <https://ttssh2.osdn.jp/>

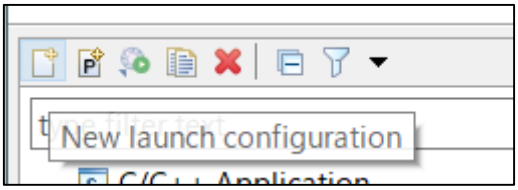
## 5.2 プログラムのデバッグ実行

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) STM32CudeIDE のメニューからメニュー「Run」→「Debug configurations」を選択し、開いたダイアログから項目「STM32 Cortex-M C/C++ Application」を選択する。



(2) 「New configuration」ボタンを押し構成を追加する。

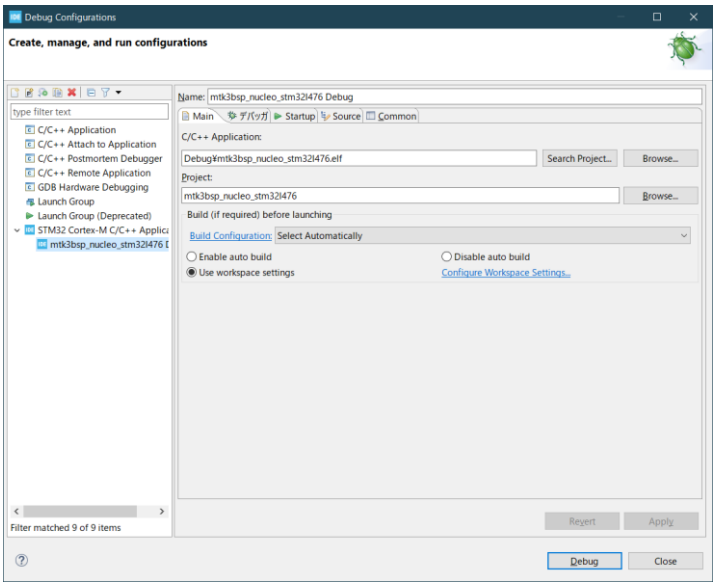



(3) 追加した構成を選択し以下の設定を行う。その他の設定についても必要に応じて変更すること。

- 「Main」タブ

  C/C++ Application：ビルドした ELF ファイル

  Project：前項で作成したプロジェクトを指定




- 「デバッガー」タブ

  デバッガプローブ：「ST-LINK (ST-LINK GDB server)」を選択

- 「Startup」タブ

  Set breakpoint at：　「usermain」を入力

(4) デバッグ開始

「Debug」ボタンを押すとプログラムが実機に転送され、STM32CubeIDE 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)で停止しする。

(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 Resume 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（STM32L4723 Nucleo-64）上の LED（LD2）を点滅させる。

また、サンプル・プログラムはデバッグ出力に「User program started」の文字列を出力する。パソコンでターミナルエミュレータを実行していれば、その画面上に文字列が表示される。

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、STM32CubeIDE 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