μT-Kernel3.0
STM32L4 IoT-Engine 向け
開発環境マニュアル

- STM32CubeIDE 編 -

Version.01.00.00

2021.05.18

更新履歴

版数(日付)	内　容
1.00.00 (2021.05.18)	● 初版

目次

1. 概要

本書は以下の対象においてμT-Kernel 3.0のアプリケーション開発を行うまでの手順を説明する。

以下に本書の対象を記す。

1.1 対象とするマイコンとOS

開発対象のマイコンおよびOSは以下である。

分類	名称	備考
マイコン	STM32L486VG (ARM Cortex-M4コア)	STマイクロエレクトロニクス製
OS	μT-Kernel3.00.04	TRONフォーラム
実機	STM32L4 IoT-Engine	UCテクノロジー製

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 v1.6.1	STMicroelectronics https://www.st.com/ja/development-tools/stm32cubeide.html

※ 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。

2. 開発環境の準備

μT-Kernel 3.0 のアプリケーション開発を行うにあたり、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

開発するμT-Kernel 3.0 のアプリケーションについて、STM32CudeIDE のプロジェクトを作成する。
TRON フォーラムでは、基本となるプロジェクトを GitHub にて公開している。このプロジェクトを使用することにより、容易にアプリケーション開発のプロジェクトを作成することができる。

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にて、各開発環境のプロジェクトが公開されている。

https://github.com/tron-forum/mtk3_devenv

プロジェクト一式は、以下のいずれかの方法で取得できる。

- プロジェクト一式を Zip ファイルとして入手する場合

master ブランチの archives ディレクトリに各プロジェクトの zip ファイルが格納されている。本プロジェクトの Zip ファイル名は「mtk3_stm32cube_stm32l4.zip」である、この

Zip をダウンロードし、任意の場所に展開する。

- プロジェクト一式を Git のクローンとして入手する場合

プロジェクト毎にブランチが設けられている。本書の対象開発環境のブランチは「stm32cube/stm32l4」である。

以下の Git コマンドを実行することにより、レポジトリのクローンが取得できる。

```
git clone -b stm32cube/stm32l4 --recursive https://github.com/tron-forum/mtk3_devenv.git
```

なお、レポジトリ中のμT-Kernel 3.0 のソースコードはサブモジュールとして、TRON フォーラムの GitHub の以下の URL のμT-Kernel 3.0 レポジトリに紐づけられている。

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

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は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
mtkernel_3	μT-Kernel 3.0 ソースコードのディレクトリ
user_program	ユーザ・プログラム用のディレクトリ
.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

ディレクトリ「mtkernel_3」の内容は、TRON フォーラムから公開されているμT-Kernel 3.0 のソースコードと同一である。

ディレクトリ「user_program」に作成するユーザ・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 user_main.c ファイルが格納されている。

その他のディレクトリやファイルは、STM32CudeIDE のプロジェクト関係のファイルである。ユーザが直接これらのファイルにアクセスすることはない。

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「mtk3_stm32l4」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「mtk3_stm32l4」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「mtk3_stm32l4」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. ユーザ・プログラムの作成

4.1 ユーザ・プログラムのソースファイル

本プロジェクトでは、ユーザ・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「user_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 user_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。
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」の違いは最適化である。通常、デバッグ時は「Debug」を使用する。なお、ワーキング・セットの各種設定は、ユーザ・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「mtk3_stm32l4.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は、実機環境により異なる。
本項では、例として以下の実機環境での手順を説明する。

項目	内容
実機ハードウェア（実行ボード）	IoT-Engine Starter Board
デバッガ（JTAG エミュレータ）	SEGGER J-Link

5.1 STM32CudeIDE によるプログラムの実行

(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：「SEGGER J-LINK」を選択
  device：「**STM32L486VG**」を入力

- 「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実行され、前項において設定したブレークポイントで停止する。

以上