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

- Eclipse 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	Eclipse 4.19.0(2021-03)	Eclipse Foundation https://www.eclipse.org/

※ 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。

(1) Eclipse のインストール
(2) クロス開発プラグインのインストール
(3) GCC Toolchain のインストール

以降、それぞれについて説明をしていく。なお、すでに各ソフトがインストール済みの場合はこの手順は不要である。

2.1 Eclipse のインストール

Eclipse および C/C++開発用プラグイン CDT、また Eclipse を実行するための Java 実行環境のインストールを行う。
これら必要なソフトウェアおよび日本語化プラグインなどをまとめたパッケージ Pleiades All in One を使用することにより、一括でインストールすることができる。
なお、Pleiades All in One を使用せずに、ここに各ソフトウェアをインストールすることも可能である。本書では Pleiades All in One を用いた方法を説明する。

Pleiades All in One は以下からダウンロードする。

MergeDoc Project https://mergedoc.osdn.jp/

本稿作成時に検証したバージョンは以下の通り。

Pleiades All in One リリース 2021-03 Full Edition
pleiades-2021-03-cpp-win-64bit-jre_20210329.zip

ダウンロードした zip ファイルを任意の場所に展開する。
Eclipse の実行は、zip ファイルを展開したディレクトリ中の以下のプログラムを起動する。

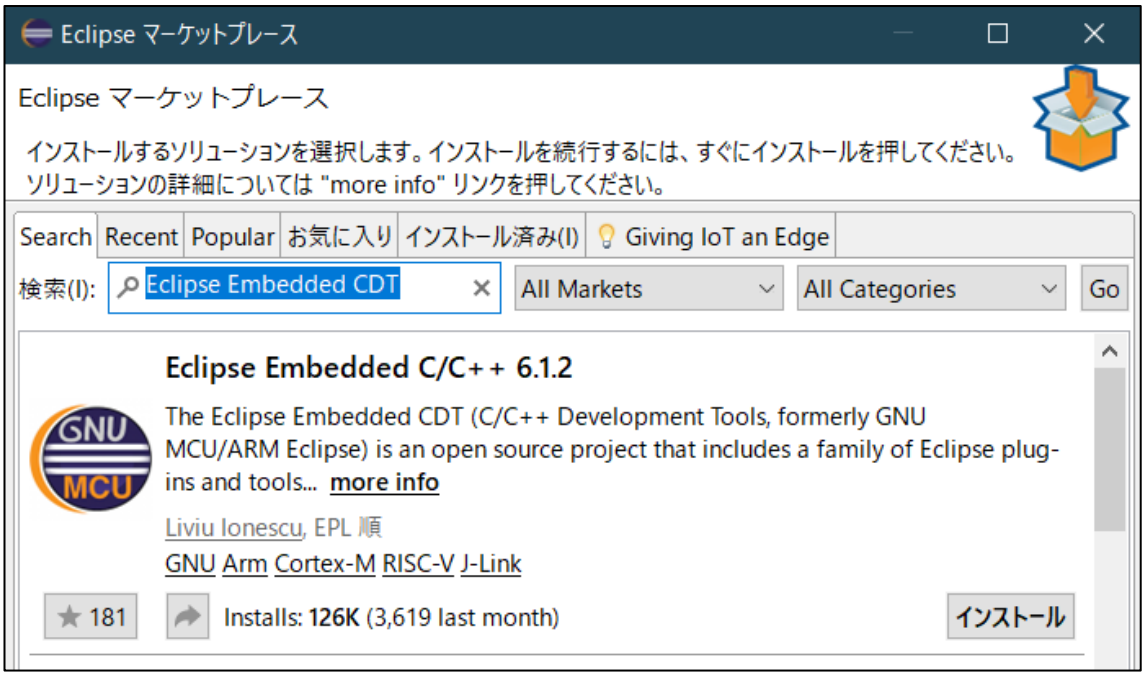
<展開したディレクトリ>¥eclipse¥eclipse.exe

Eclipse の初回起動時、指示に従いワークスペースを作成する。ワークスペースは Eclipse の各種設定などが保存される可能的な作業用ディレクトリである。

2.2 クロス開発プラグインのインストール

Eclipse にクロス開発プラグインをインストールする。

(1) Eclipse を起動し、メニュー「ヘルプ」→「Eclipse マーケットプレース」を選択する。
(2) 開いたダイアログの検索欄に「Eclipse Embedded CDT」を入力し、「Go」を押して実行する。
(3) 検索結果として「Eclipse Embedded C/C++」が表示されるので、それをインストールする。



2.3 GCC Toolchain のインストール

C コンパイラ等開発ツール一式をインストールする。

(1) C コンパイラのインストール

GCC コンパイラー式を以下からダウンロードし、任意の場所に展開する。。

GNU Arm Embedded Toolchain

https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm

(2) 開発ツールのインストール

GCC toolchain を使用するためのツール一式（make など）を以下からダウンロードし、任意の場所に展開する。

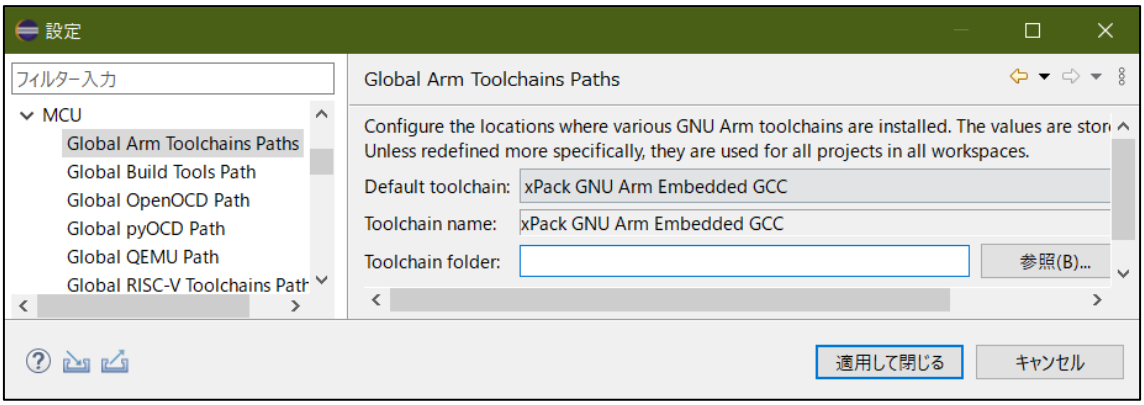
xPack Windows Build Tools

https://github.com/xpack-dev-tools/windows-build-tools-xpack/releases

(3) 実行パスの設定

GCC を展開したディレクトリ内の¥bin ディレクトリおよび、xPack Windows Build Tools を展開したディレクトリ内の¥bin ディレクトリを、OS のコマンド実行パスに設定する。

または、Eclipse の設定にて、「Arm Toolchains Path」および「Build Tools Path」にそれぞれのディレクトリを設定する。



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、Eclipse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_eclipse_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「eclipse/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 eclipse/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	Eclipse の設定ファイル
.cproject	Eclipse のプロジェクトファイル

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、Eclipse 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

入手したプロジェクトを Eclipse の開発環境に以下の手順で取り込む。

(1) メニュー「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」で「参照」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) 「プロジェクト」に「mtk3_stm32l4」が表示されるので、それをチェックする。
(5) 「完了」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、Eclipse 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「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. 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

Eclipse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「プロジェクト」→「ビルド構成」→「アクティブにする」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 SEGGER J-Link Software のインストール

(1) SEGGER J-Link Software を次の Web サイトからダウンロードする。

SEGGER https://www.segger.com/

サイトの「Download」→「J-Link/J-Trace」を選択し、 J-Link Software and Documentation Pack をダウンロードする。以下のインストーラがダウンロードされる（バージョンは変更される可能性がある）。

JLink_Windows_V656a.exe

(2) ダウンロードしたインストーラを実行し、SEGGER J-Link Software をインストールする。

5.2 Eclipse によるプログラムの実行

(1) Eclipse 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「GDB SEGGER J-Link Debugging」を選択する。

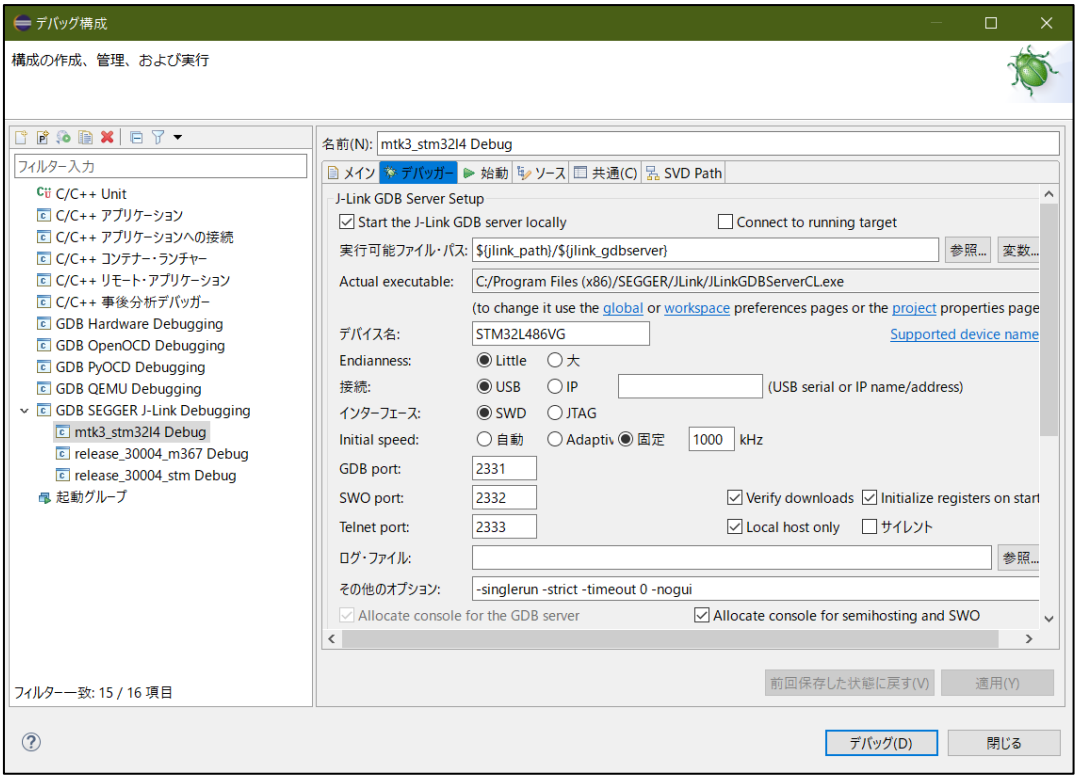
(2) 「新規構成」ボタンを押し、「GDB SEGGER J-Link Debugging」に構成を追加する。

(3) 追加した構成を選択し、「構成の作成、管理、実行」画面にて以下の設定を行う。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
  名前：(任意) を入力
  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

- 「デバッガー」タブ
  デバイス名：「**STM32L486VG**」を入力

- 「始動」タブ
  Set breakpoint at： usermain



(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