

**$\mu$ T-Kernel3.0**  
**TX03 M367 IoT-Engine 向け**  
**開発環境マニュアル**

**－ SEGGER Embedded Studio 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 .....	4
1.1	対象とする OS およびハードウェア.....	4
1.2	対象とする開発環境.....	4
2.	開発環境の準備 .....	5
2.1	Embedded Studio のインストール .....	5
2.2	CPU サポートパッケージのインストール .....	5
3.	プロジェクトの作成.....	6
3.1	GitHub からのプロジェクトの入手 .....	6
3.2	プロジェクトのファイル構成.....	7
3.3	プロジェクトのオープン.....	7
3.4	プロジェクトのビルドの確認.....	7
4.	ユーザ・プログラムの作成.....	8
4.1	ユーザ・プログラムのソースファイル.....	8
4.2	コンフィギュレーションの選択.....	8
4.3	実行プログラムのビルド.....	8
5.	実機でのプログラム実行.....	9

## 1. 概要

本書は以下の対象において  $\mu$ T-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 OS およびハードウェア

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	TX03 シリーズ M360 グループ TMPM367FDFG (ARM Cortex-M3 コア)	東芝デバイス & ストレージ製
OS	$\mu$ T-Kernel 3.00.04	TRON フォーラム
実機	TX03 M367 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	Embedded Studio for ARM Release 5.32a	SEGGER <a href="https://www.segger.com/">https://www.segger.com/</a>

※ バージョン番号は確認した最新版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) Embedded Studio のインストール
- (2) CPU サポートパッケージのインストール

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 Embedded Studio のインストール

以下から使用する PC の OS に合わせて、ARM 向けの Embedded Studio をダウンロードする。

#### Embedded Studio のダウンロードページ

<https://www.segger.com/downloads/embedded-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。特に問題が無ければデフォルトの設定でインストールを行えばよい。

### 2.2 CPU サポートパッケージのインストール

Embedded Studio を実行し、開発対象とする CPU に応じたサポートパッケージを以下に手順でインストールする。

- (1) メニュー「Tools」→「Package Manager」を選択する。
- (2) ダイアログの検索欄に「TMPM3」と入力すると、検索結果に「TMPM3 CPU Support Package」が表示される。
- (3) 「TMPM3 CPU Support Package」を右クリックし、ポップアップしたメニューから「Install Selected Packages」を選択する。以降、指示に従いインストールを進める。

### 3. プロジェクトの作成

開発する  $\mu$ T-Kernel 3.0 のアプリケーションについて、Embedded Studio 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](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 archive 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\_seggeres\_m367.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「segger\_es/m367」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 segger_es/m367 --recursive https://github.com/tron-forum/mtk3_devenv.git
```

なお、レポジトリ中の  $\mu$ T-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  $\mu$ T-Kernel 3.0 レポジトリに紐づけられている。

[https://github.com/tron-forum/mtkernel\\_3](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	$\mu$ T-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
mtk3_m367.emProject	Embedded Studio 用プロジェクト・ファイル
その他のファイル	Embedded Studio 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  $\mu$ T-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。

その他のファイルは、Embedded Studio の各種ファイルである。ユーザが直接これらのファイルにアクセスする必要はない。

### 3.3 プロジェクトのオープン

Windows の場合は、プロジェクトのディレクトリ中の「mtk3\_m367.emProject」ファイルをエクスプローラから開くと、Embedded Studio が起動し、本プロジェクトがオープンされる。

### 3.4 プロジェクトのビルドの確認

プロジェクトの  $\mu$ T-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\_m367」が選択されている状態で、メニュー「Build」→「Build mtk3\_m367」を選択する。

プログラムのビルドが実行され、以下が表示されれば、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 コンフィギュレーションの選択

Embedded Studio は初期状態では、「Release」と「Debug」の二つのコンフィギュレーションが作られている。

コンフィギュレーションは、メニュー「Build」→「Set Active Build Configuration」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、コンフィギュレーションの各種設定は、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
Optimization Level	None	Level2 balance
Debugging Level	Level 3	Level 1

### 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「Build」→「Build mtk3\_M67」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\_m367.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に「Output」ディレクトリが生成られ、その中の以下に生成される。

Debug コンフィギュレーションの場合

<プロジェクトのディレクトリ>%Output%Debug%Exe%mtk3\_m367.elf

Release コンフィギュレーションの場合

<プロジェクトのディレクトリ>%Output%Release%Exe%mtk3\_m367.elf



## 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。  
本プロジェクトでは、JTAG エミュレータ SEGGER J-Link を使用したデバッグ実行環境が設定されている。以下に動作を確認した実機環境を記す。

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

メニュー「Build」→「Build and Debug」を選択すると、プログラムは実機に転送され、デバッグ実行が開始する。

以上