μT-Kernel3.0 BSP マニュアル

- Raspberry Pi Pico編 -

Version.01.00.00

2022.11.30

更新履歴

版数(日付)	内　容
1.00.00 (2022.11.30)	● 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	RP2040 (ARM Cortex-M0+コア)	Raspberry Pi Foundation
OS	μT-Kernel3.00.06	TRON フォーラム ※1
実機 (マイコンボード)	Raspberry Pi Pico	Raspberry Pi Foundation

※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

統合開発環境には Eclipse IDE for Embedded C/C++ Developers（以下 Eclipse Embedded IDE）を使用する。
開発を行うホスト PC の OS は Windows とする。確認は Windows 11 にて行った。

分類	名称	備考
C コンパイラ	The xPack GNU Arm Embedded GCC	
開発ツール	xPack Windows Build Tools	
統合開発環境	Eclipse IDE for Embedded C/C++ Developers	Eclipse Foundation

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 Raspberry Pi Pico 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	UART0	sera
	UART1	serb
A/D 変換デバイスドライバ	ADC	adca
I^2C 通信デバイスドライバ	I2C0	iica
	I2C1	iicb

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.1.00.5)	TRON フォーラム ※2 TEF033-W007-221007

※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 C 開発ツールのインストール

C コンパイラなどの開発ツール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入手し、Web ページの指示に従いインストールする。

The xPack GNU Arm Embedded GCC

https://xpack.github.io/arm-none-eabi-gcc/

本稿作成時に検証したバージョンは以下の通り。

xpack-arm-none-eabi-gcc-11.3.1-1.1

(2) 開発ツールのインストール

開発ツール一式（make など）を以下から入手し、Web ページの指示に従いインストールする。

xPack Windows Build Tools

https://xpack.github.io/windows-build-tools/

本稿作成時に検証したバージョンは以下の通り。

xPack Windows Build Tools v4.3.0-1

2.2 Eclipse Embedded IDE のインストール

使用する PC の OS に合わせて以下から Eclipse Embedded IDE をダウンロードする。

Eclipse Packages のダウンロードページ

https://www.eclipse.org/downloads/packages/

本ソフトの動作検証には以下の Eclipse Embedded IDE の Package を使用した。

eclipse-embedcpp-2022-09-R-win32-x86_64

3. プロジェクトの作成

Eclipse Embedded IDE 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。Raspberry Pi Pico の BSP のブランチ名は「pico_rp2040」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。

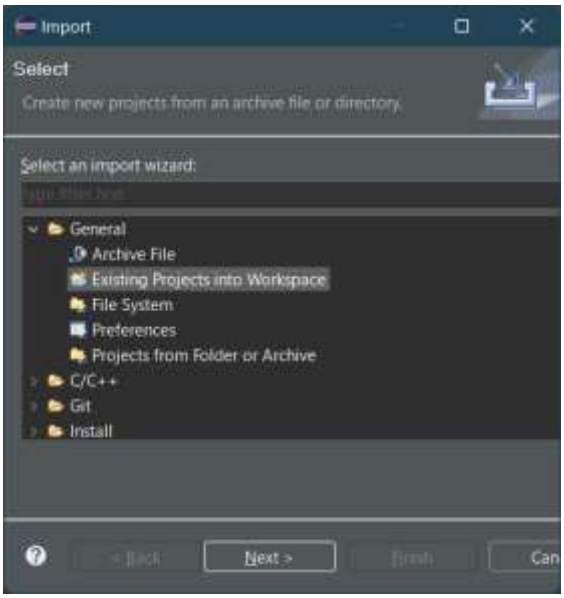
ディレクトリまたはファイル名	内容
app_program	アプリケーション・プログラム用のディレクトリ
build_make	Make 構築ディレクトリ（BSP では使用しない）
config	コンフィギュレーション
device	デバイスドライバ
docs	ドキュメント
etc	リンカファイル等
include	各種定義ファイル
kernel	μT-Kernel 本体
lib	ライブラリ
.settings	Eclipse Embedded IDE の設定ファイル
.cproject	Eclipse Embedded IDE のプロジェクトファイル
その他ファイル	Eclipse Embedded IDE 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

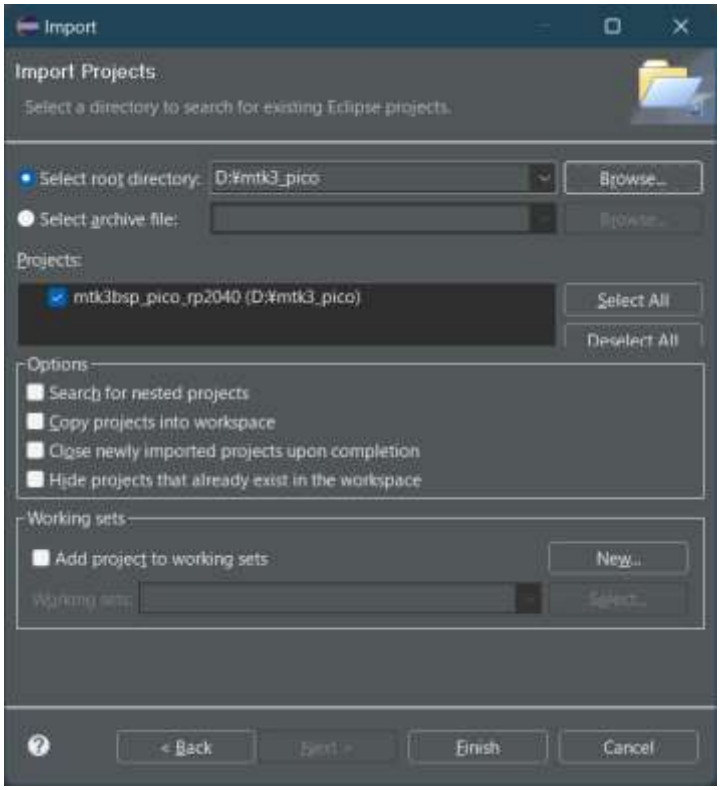
前項で入手したプロジェクトを Eclipse Embedded IDE の開発環境に以下の手順で取り込む。

(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。

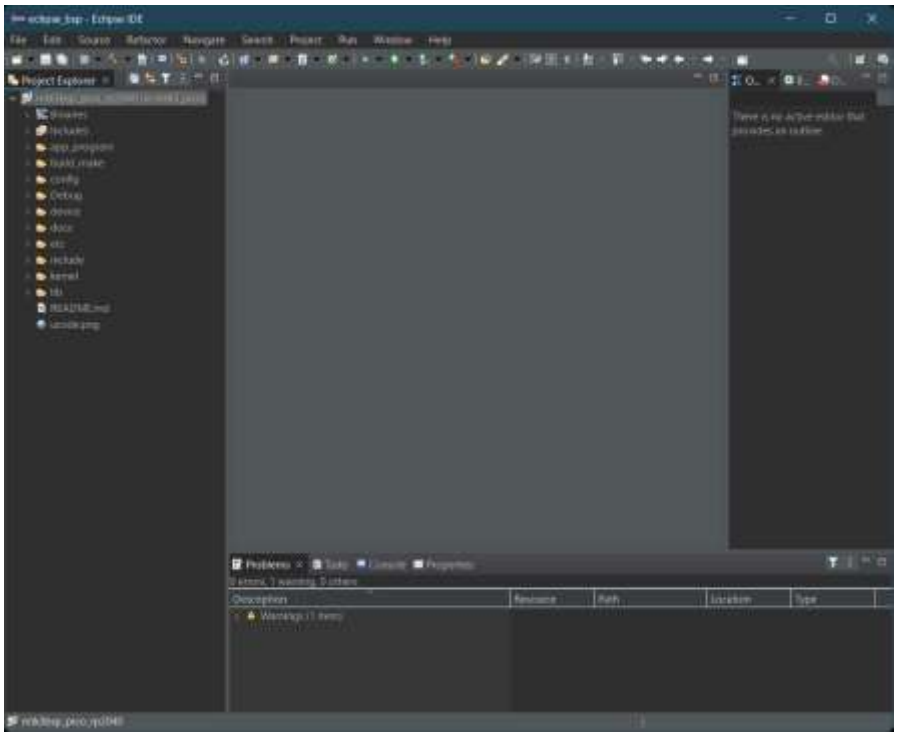


(3) 「Select root directory」で「Browse」ボタンを押し、前項で入手したプロジェクトのディレクトリを指定する。

(4) 「Projects」に「mtk3bsp_pico_rp2040」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、Eclipse Embedded IDE のプロジェクト・エクスプローラに「mtk3bsp_pico_rp2040」が表示される。

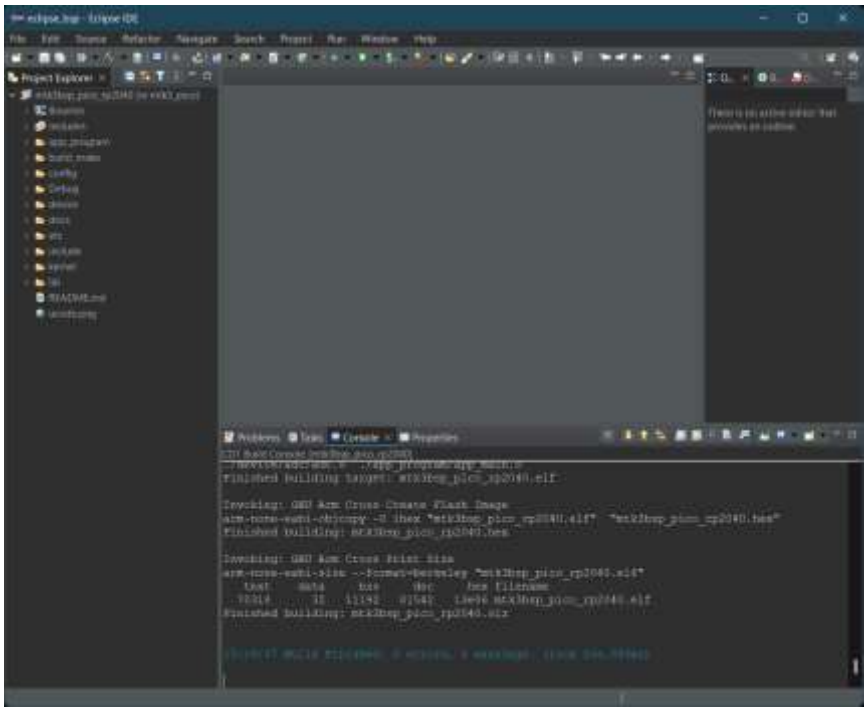


3.4 プロジェクトのビルドの確認

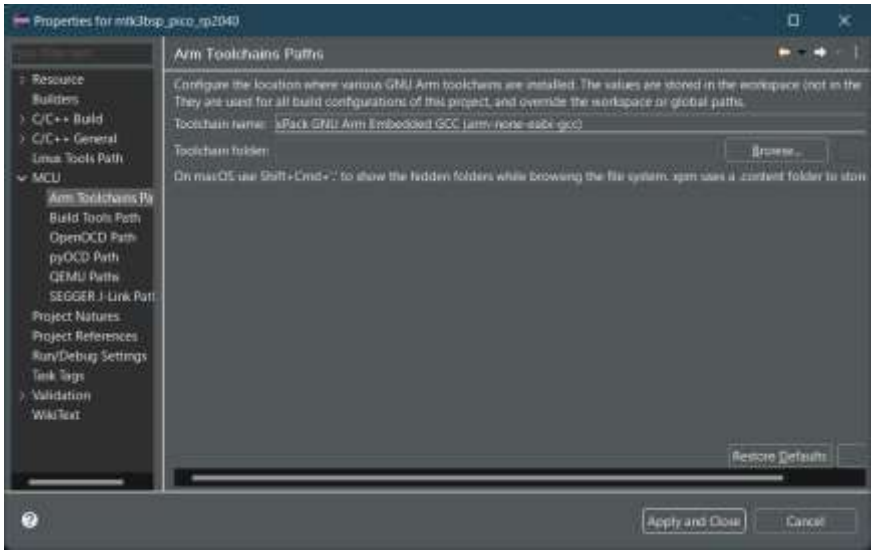
プロジェクトのμT-Kernel 3.0のプログラムがビルド出来ることを確認する。

プロジェクト「mtk3bsp_pico_rp2040」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。



ビルドでエラーが発生する場合は、C コンパイラおよび開発ツールのパスが設定されていない可能性がある。プロジェクト「mtk3bsp_pico_rp2040」が選択されている状態で、メニュー「File」→「Properties」を選択し、項目「MCU」の「ARM Toolchains Path」および「Build Tool Path」のパスの設定を行う。



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

Eclipse Embedded IDE 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_pico_rp2040.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

Eclipse Embedded IDE を実行しているパソコンと実機ボード（Raspberry Pi Pico）を JTAG デバッグプローブで接続する。
JTAG デバッグプローブは、Picoprobe を使用して検証している。Picoprobe については、Raspberry Pi Ltd のドキュメントを参考のこと。Picoprobe の使用に当たっては「Getting started with Raspberry Pi Pico」に記載されている OpenOCD のインストールが必要となる。

Raspberry Pi Documentation
https://www.Raspberry Pi.com/documentation/microcontrollers/

Getting started with Raspberry Pi Pico (PDF)
https://datasheets.Raspberry Pi.com/pico/getting-started-with-pico.pdf

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 Picoprobe 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)として、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。シリアルポートの設定は以下とする。

通信速度： 115200bps
データ：8bit
パリティ：none
ストップビット:1bit

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

5.2 プログラムのデバッグ実行

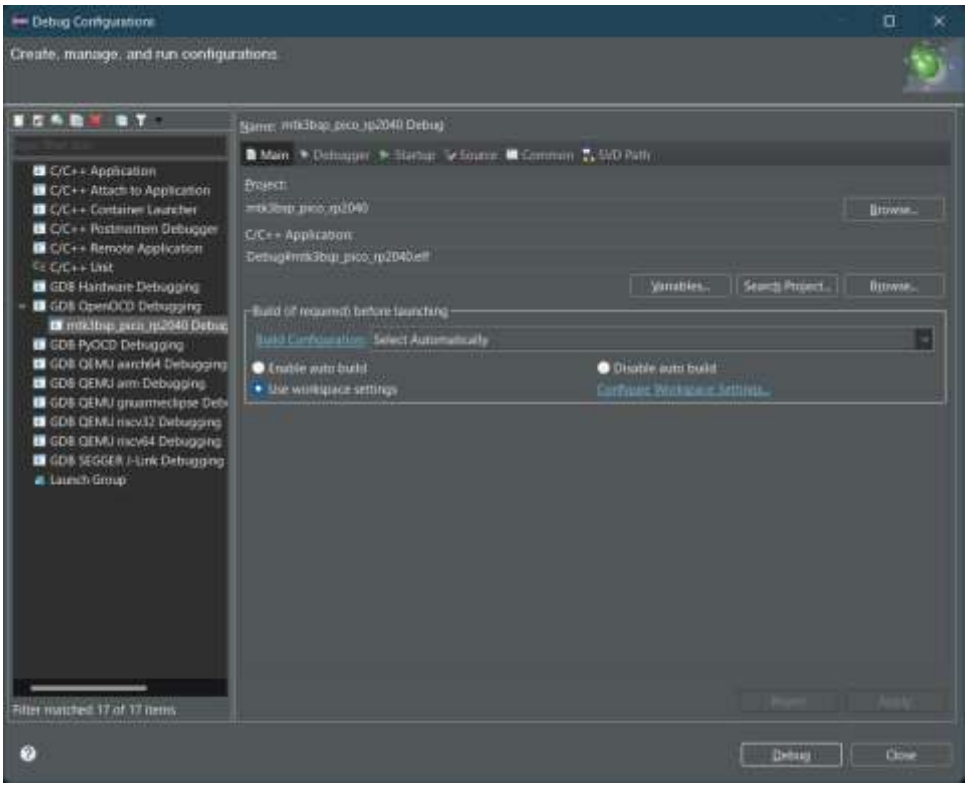
プログラムのデバッグ実行は、以下の手順で行う。なお(1)から(3)までの操作は最初に1回のみ行うだけでよい。

(1) 「4.3 実行プログラムのビルド」を行った後に、Eclipse Embedded IDE のメニューからメニュー「Run」→「Debug configurations」を選択する。

(2) 開いたダイアログから項目「GDB OpenOCD Debugging」を選択し、「New configuration」ボタンを押すと、「mtk3bsp_pico_rp2040」が新規構成として追加される。

(3) 追加した構成を選択し以下の設定を行う。すでに正しい値が設定されている場合は変更の必要はない。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

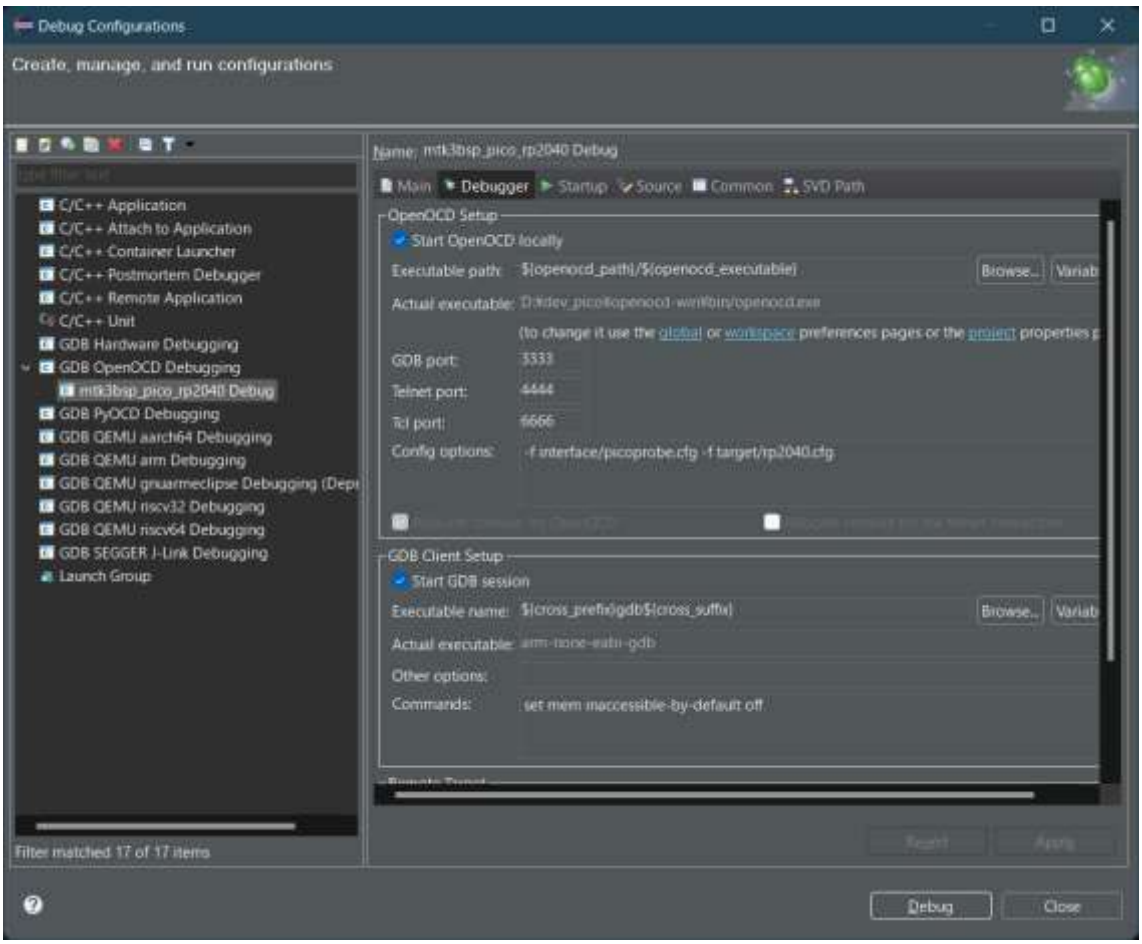


- 「Debugger」タブ
  OpenOCD の Excutable path およびおよび GDB Client の Execurable name をインストールしたツールに合わせて設定する。
  OpenOCD の Config options に以下を設定する。

  -f interface/picoprobe.cfg -f target/rp2040.cfg

  GDB Client の Commands に以下を設定する。

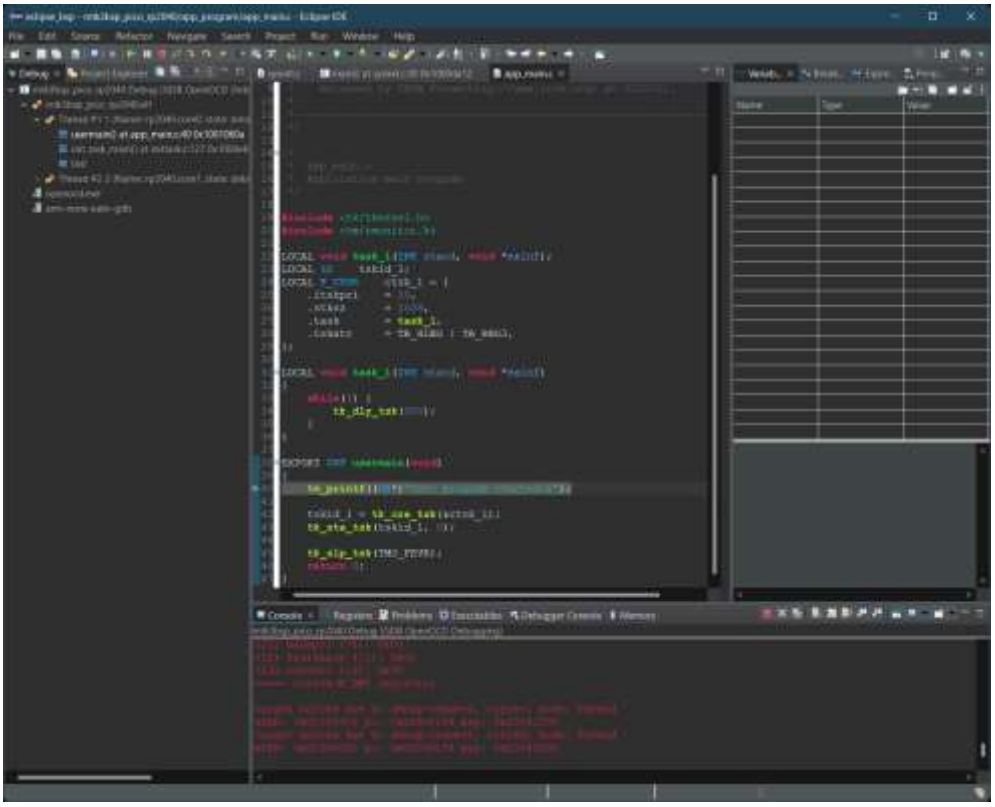
  set mem inaccessible-by-default off



- 「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、Eclipse Embedded IDE はデバッグ画面に変わる。



main 関数から再開したプログラムは、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。Resume ボタンを押下するとサンプル・プログラムが実行される。

サンプル・プログラムは、実機ボード（Raspberry Pi Pico）上の LED 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、Eclipse Embedded IDE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