Creating New Era, Together

PUYA MCU

PY32 シリーズ　クイックスタートガイド

PY32_GCC_SDE（VS Code）バージョン

目次

1. 概要

本書は、PUYA Semiconductor 社製 マイコンの開発環境構築およびビルド、Flash Write、Debug 手順を解説したセットアップマニュアルです。VS Code（Visual Studio Code）と Embedded IDE（EIDE）を使用した効率的な開発環境の立ち上げ手順を記載しています。

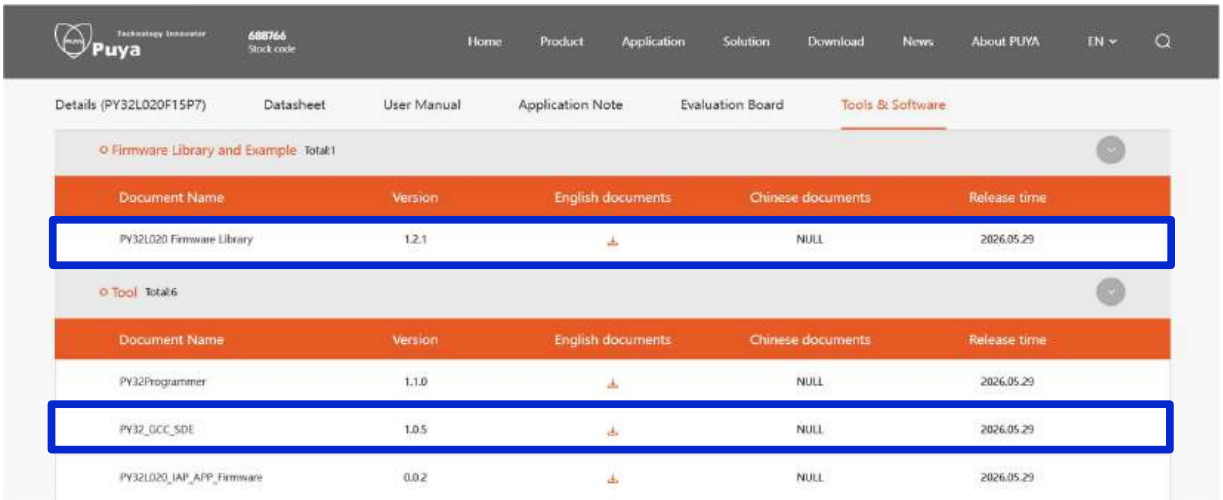
2. 事前準備（ファイルのダウンロード）

開発を開始する前に、以下の Web サイトの「Tools & Software」タブから必要なファイルをダウンロードしてください。

> **ダウンロード先 URL:**
> https://www.puyasemi.com/en/py32l020/2711.html

ダウンロード項目	ファイル名 / 内容	Version
※PY32L020 Firmware Library	PY32L020_Firmware_V1.2.1.zip （ファームウェアライブラリ・サンプルコード群）	1.2.1
PY32_GCC_SDE	PY32_GCC_SDE_V1.0.5.zip （統合開発環境セットアップパッケージ）	1.0.5

※今回は PUYA 製マイコン：PY32L020 を例に示すため、上記 BSP をダウンロード。



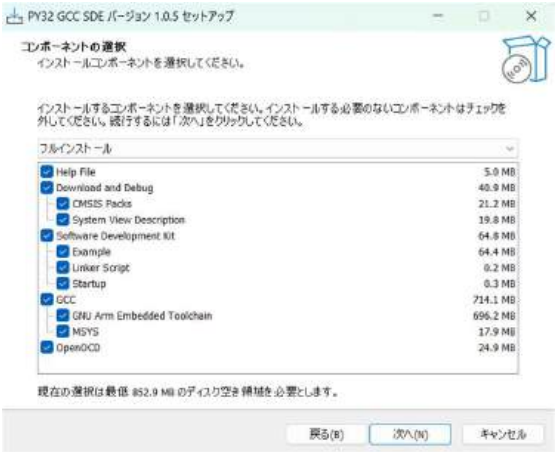
3. 開発環境のインストールおよび構築

ステップ 3-1 : PY32_GCC_SDE のインストール

3.1.1 「PY32_GCC_SDE_V1.0.5.zip」を解凍し、「PY32_GCC_SDE_1.0.5.exe」を実行します。

3.1.2 言語選択で「日本語」を選択し、インストール先フォルダを指定します。

3.1.3 インストールコンポーネントの選択画面では「フルインストール」を推奨します。

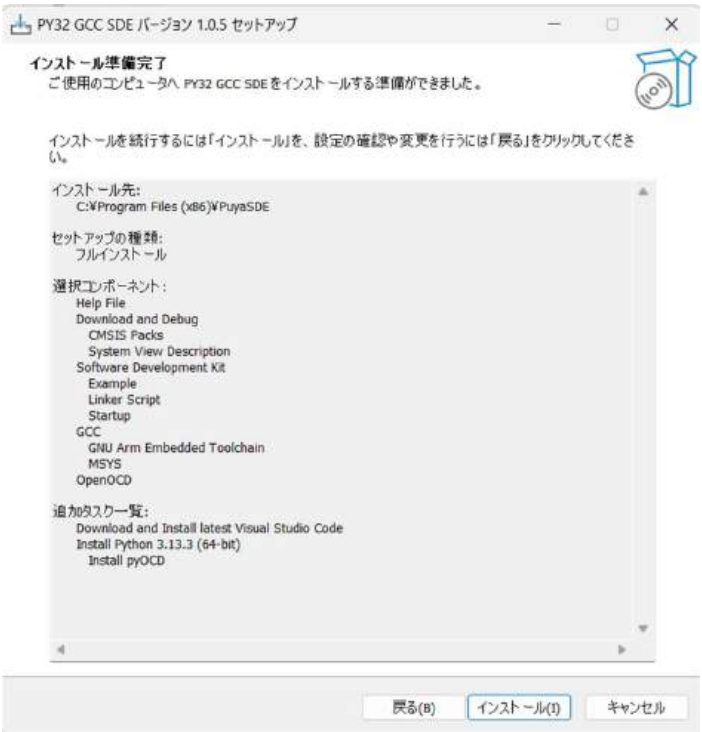


3.1.4 追加タスクの選択画面で以下の項目にチェックを入れてインストールを進めます。インストール中に追加ファイルをダウンロードするため、インターネットにつながった環境で実行してください。

・Visual Studio Code のダウンロード/インストール

・Python 3.13.3 のインストール

・pyOCD のインストール



3.1.5 続いて Microsoft Visual Studio Code のセットアップ画面に従い、ライセンスに同意して、インストールを完了させます。

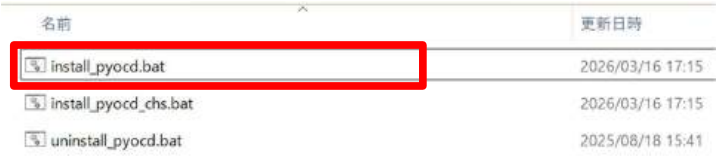
VS Code のインストール後も PY32 GCC SDE セットアップウィザードが Python などのインストールを続けるため、「VS Code を実行する」のチェックは外して完了してください。

3.1.6 PY32_GCC_SDE セットアップウィザードを完了します。

Microsoft .NET Runtime のインストール画面が出ている場合はインストールしてください。

3.1.7 以下のフォルダに生成された Batch ファイル（install_pyocd.bat）を実行して、pyOCD をインストールします。

Batch ファイル生成先：C:¥Program Files (x86)¥PuyaSDE¥bat¥install_pyocd.bat



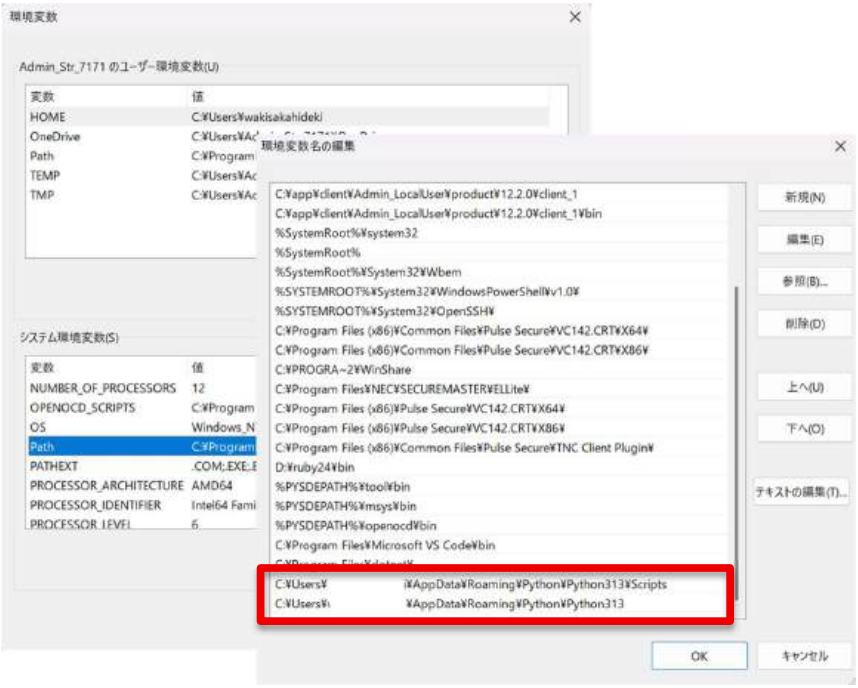
以下フォルダに pyOCD.exe を含む、python フォルダが作成されます。

Python フォルダ生成先：C:¥Users¥ユーザ名¥AppData¥Roaming¥Python

pyOCD.exe へのパスを通すため、システム環境変数に以下のパスを追加します。

・C:¥Users¥ユーザ名¥AppData¥Roaming¥Python¥Python313¥Scripts

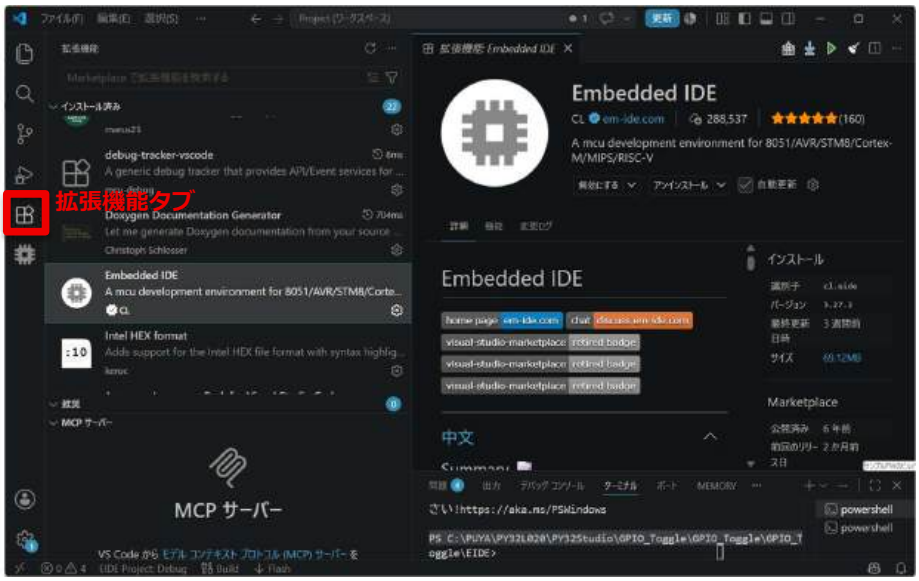
・C:¥Users¥ユーザ名¥AppData¥Roaming¥Python¥Python313



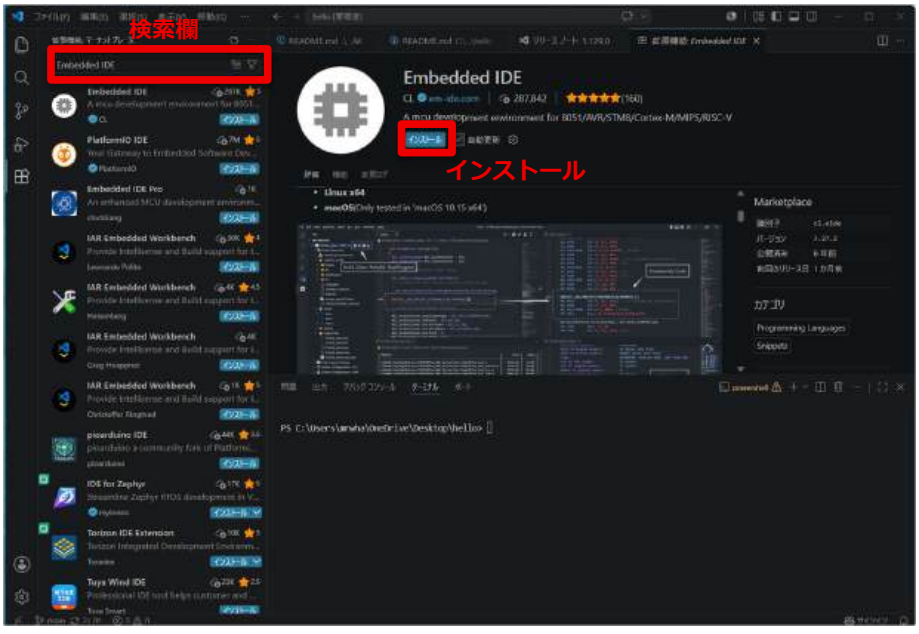
ステップ 3-2 : VS Code 拡張機能 (Embedded IDE) のインストール

3.2.1 3.1.6 でインストールした Visual Studio Code を起動します。

3.2.2 左側のサイドバーから「拡張機能（Extensions）」タブを開きます。



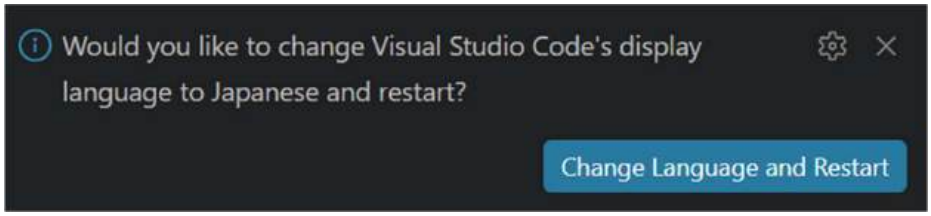
3.2.3 拡張機能検索欄に「Embedded IDE」と入力し、該当する拡張機能をインストールします。



ステップ 3-3 : VS Code 拡張機能 (Japanese Language Pack for VS Code) のインストール (Option)

3.3.1 日本語化をする場合は、ステップ 3-2 と同様に拡張機能検索欄に「Japanese Language Pack」と入力し、該当する拡張機能をインストールします。

VS Code を一度終了し、言語設定を反映させます。



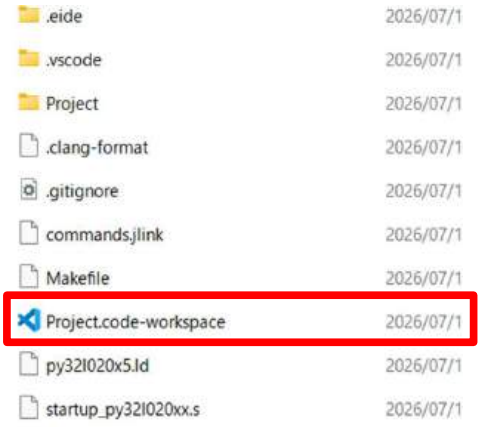
4. プロジェクトの読み込みとビルド（Build）

4.1 「PY32L020_Firmware_V1.2.1.zip」を解凍します。

4.2 目的のサンプルプロジェクトを選択します。今回は GPIO_Toggle をサンプルとして使用します。

（例:Example の GPIO_Toggle を選択：.../ Projects¥PY32L020-STK/Example/GPIO/GPIO_Toggle）

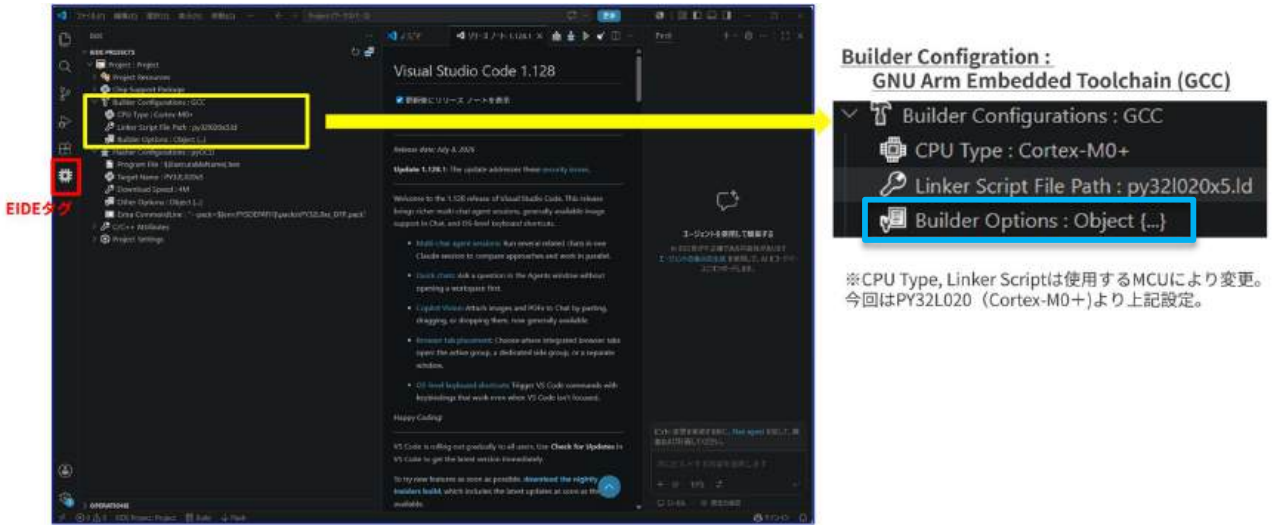
4.3 プロジェクトフォルダ内の「EIDE」フォルダにある VS Code ワークスペースファイル（Project.code-workspace）を開きます。



「制限モード」になっている場合は、左下の青い「制限モード」のボタンをクリックし、信頼済みワークスペースの「信頼する」をクリックします。

4.4 VS Code 左側の「EIDE」タブを開き、Builder Configurations：GCC

（CPU Type: Cortex-M0+、Linker Script 等）の設定が正しく反映されているか確認します。

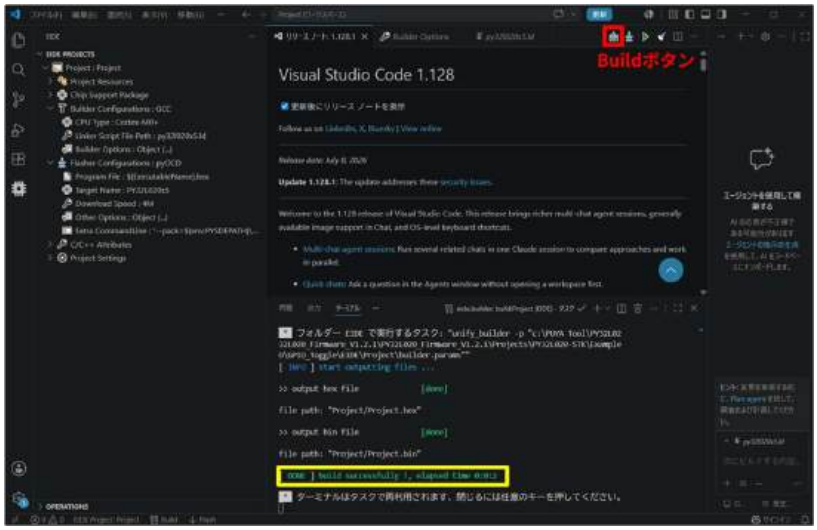


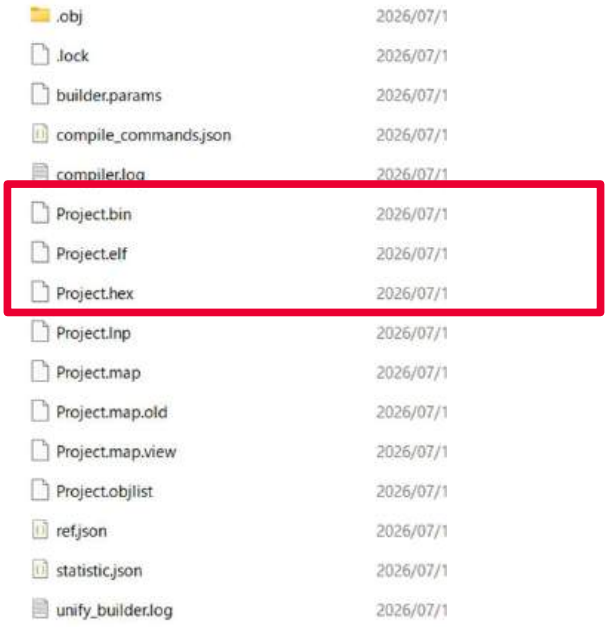
尚、Builder Options：Object｛...｝にて Compiler,Assembler,Linker などの各種 Build 設定が可能です。

4.5 画面上部または EIDE パネル内の「Build」ボタンを押してビルドを実行します。

4.6 ビルドが正常に完了すると、ターミナルに [DONE] build successfully ! と表示され、

EIDE/Project フォルダ内に ROM ファイル（.hex / .bin 等）が生成されます。





5. トラブルシューティング（build エラー対処法）

⚠ Build エラー（.s ファイルの自動書き換え）が発生する場合

【現象】

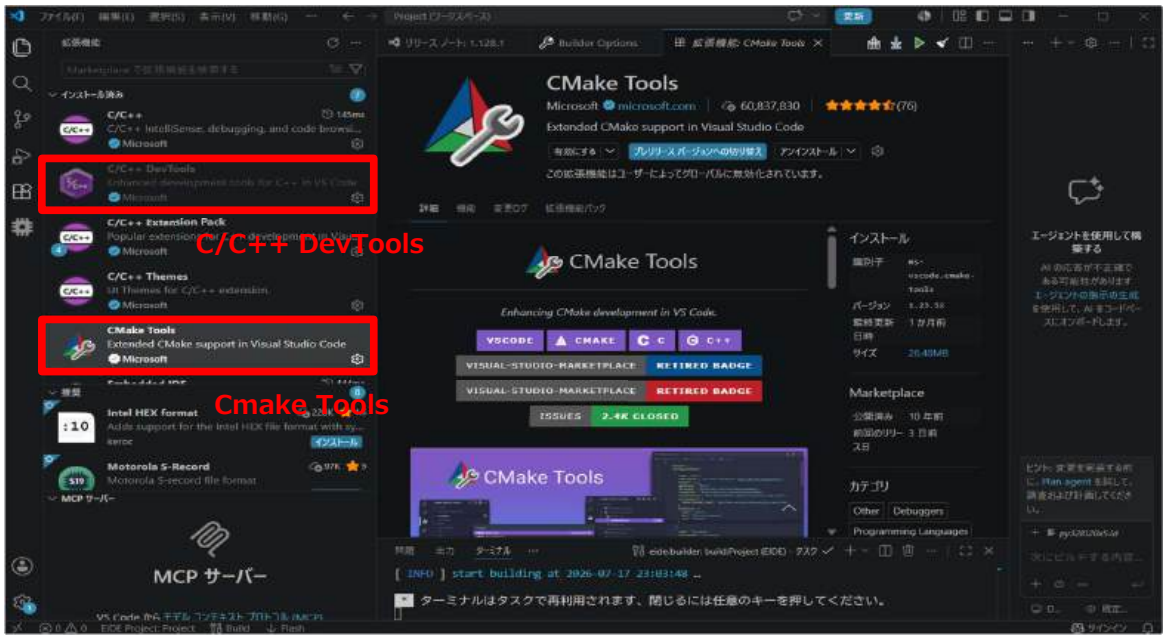
ビルド実行時にスタートアップファイル「startup_py32l020xx.s」などの .s ファイルが拡張機能により自動的に書き換えられ、ビルドエラーが発生する場合があります。

【対処法】

VS Code の拡張機能タブを開き、以下の拡張機能を「無効化（Disable）」してください。

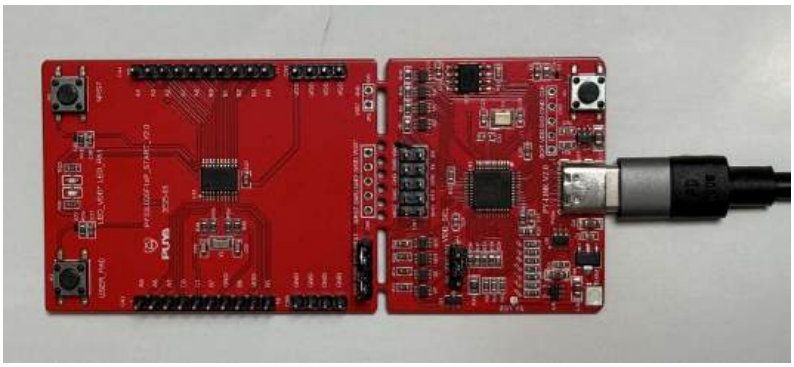
・C/C++ DevTools

・CMake Tools



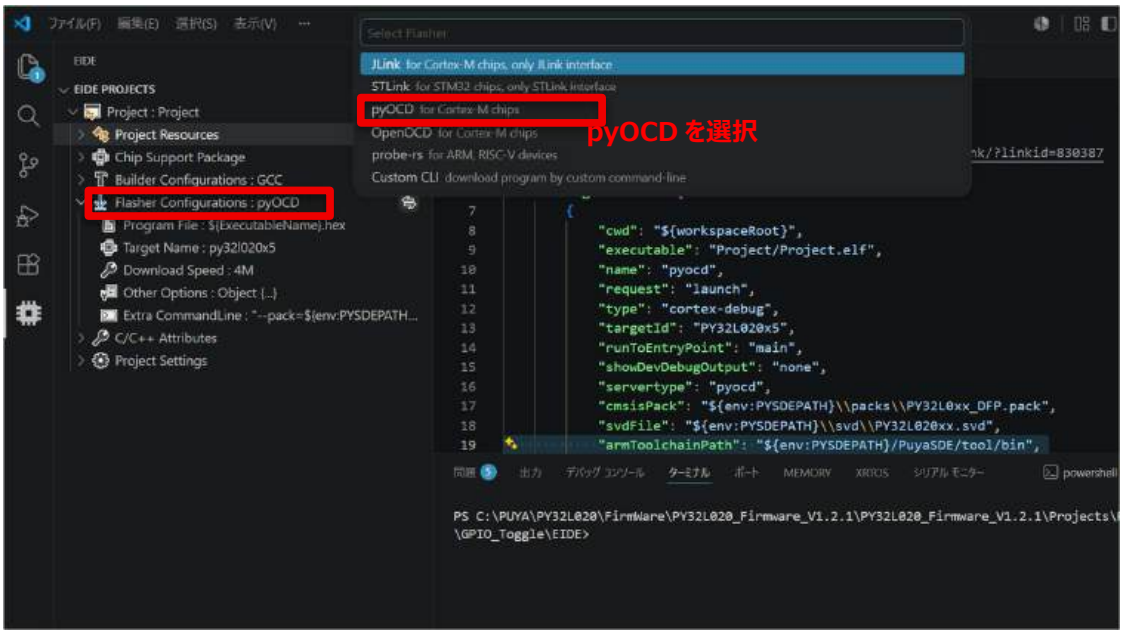
6. Flash ROM Write

6.1 VS Code をインストールした PC に PY32L020F1xP_START_V2.0 ボードを USB ケーブル(Type-C コネクタ)で接続します。



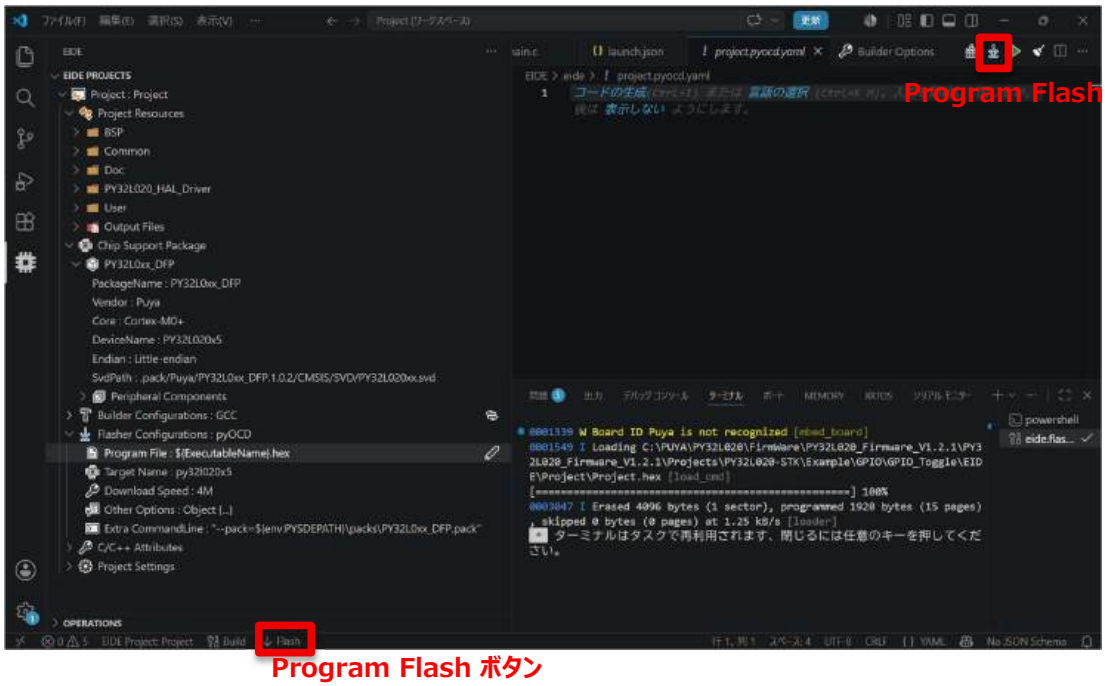
6.2 Flash ROM Write は VS Code の EIDE PROJECTS の Flasher Configurations：pyOCD を選択。

(pyOCD は、Arm® Cortex®-M マイコン向けのプログラミングとデバッグを行うための Python 製オープンソースライブラリ)



6.3 build により生成された ROM ファイル（.hex ファイル）を Write します。

画面上部または EIDE パネル内の「Program Flash」ボタンを押して Write を実行します。



6.4 正常に書き込みが完了するとターミナル画面に表示される「プログレスバー」が 100%になり、programmed サイズ（ページ単位）が表示されます。

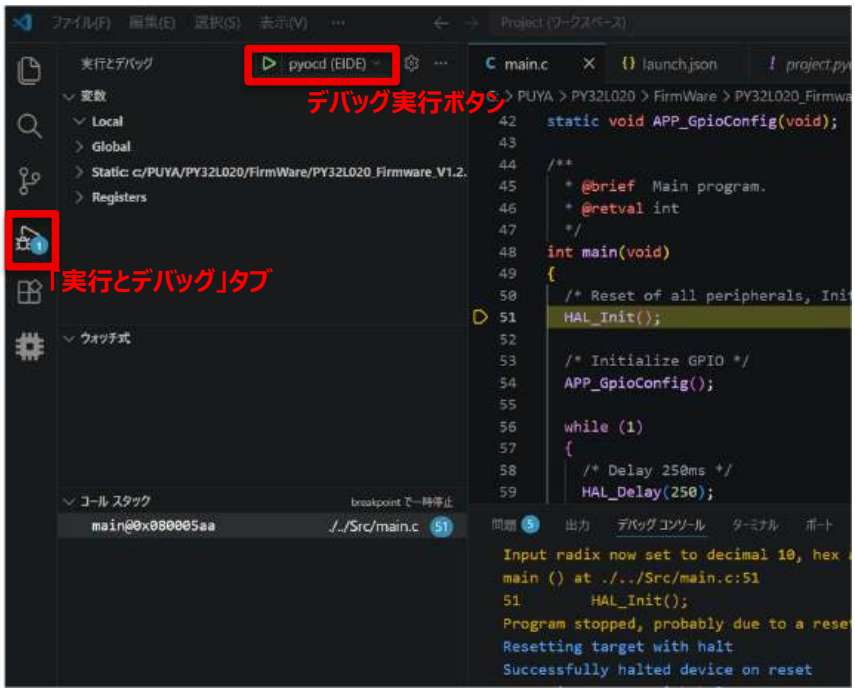
7. プロジェクトのデバッグ（Debug）

7.1 デバッグも VS Code で実施します。

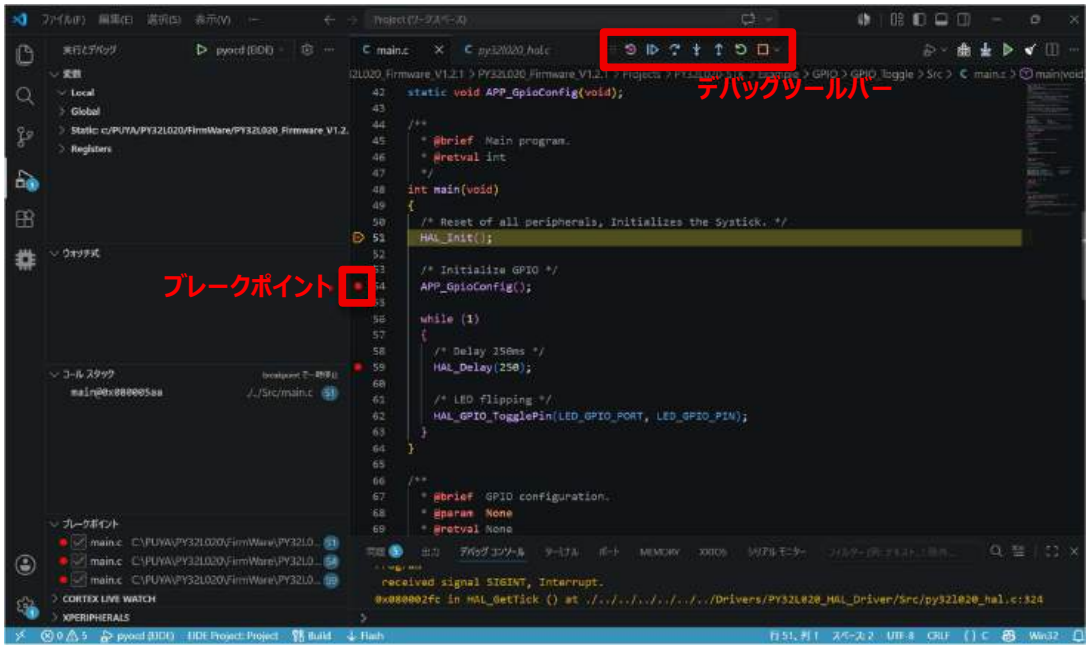
VS Code 左側の「実行とデバッグ」タブを開き、

「デバッグ実行ボタン」のデバッガは「pyocd(IDE)」を選択します。

選択後、「デバッグ実行ボタン」を押し、デバッガを起動します。



7.2 正常起動するとデバッグツールバーが表示されます。



7.3 基本動作説明

7.3.1 ブレークポイント

コードの処理を止めたい行数の左側余白部分を押すと「ブレークポイント(赤丸)」を設置できます。
(再度ブレークポイントをクリックすると解除されます)

デバッグ実行ボタンを押した後、ブラウザで該当のファイルにアクセスする(再読み込み)とブレークポイントの位置で処理が止まります。

7.3.2 ステップ実行

デバッグツールバーは処理が停止したプログラムのステップ実行が可能(次の行に進む・関数またはメソッド内の処理への出入り・デバッグを最初から始める・デバッグの停止等)で、1 行ずつ実行することで変数の値が追えるので処理の確認、エラーやバグの修正が捗ります。

8. トラブルシューティング（Debug エラー対処法）

⚠ Debug エラー（Failed to launch PyOCD GDB Server: Timeout.）が発生する場合

【現象】

GDB（GNU Debugger）Server が正常起動できず、Timeout エラーが発生する場合があります。

【対処法】

以下①、②、③の対処法を実施します。

対処法①　Windows の環境変数に追加。

下記環境変数を追加します。

変数名：　HOME

変数値：　C:¥Users¥ユーザ名

対処法②　launch.json を修正。

（launch.json はプログラムデバッグ時の動作を定義するための設定ファイル）

launch.json に　“overrideGDBServerStartedRegex”:“GDB server listening on port”の記載がある場合、コメントアウト(削除)する。

対処法③　launch.json の追加。

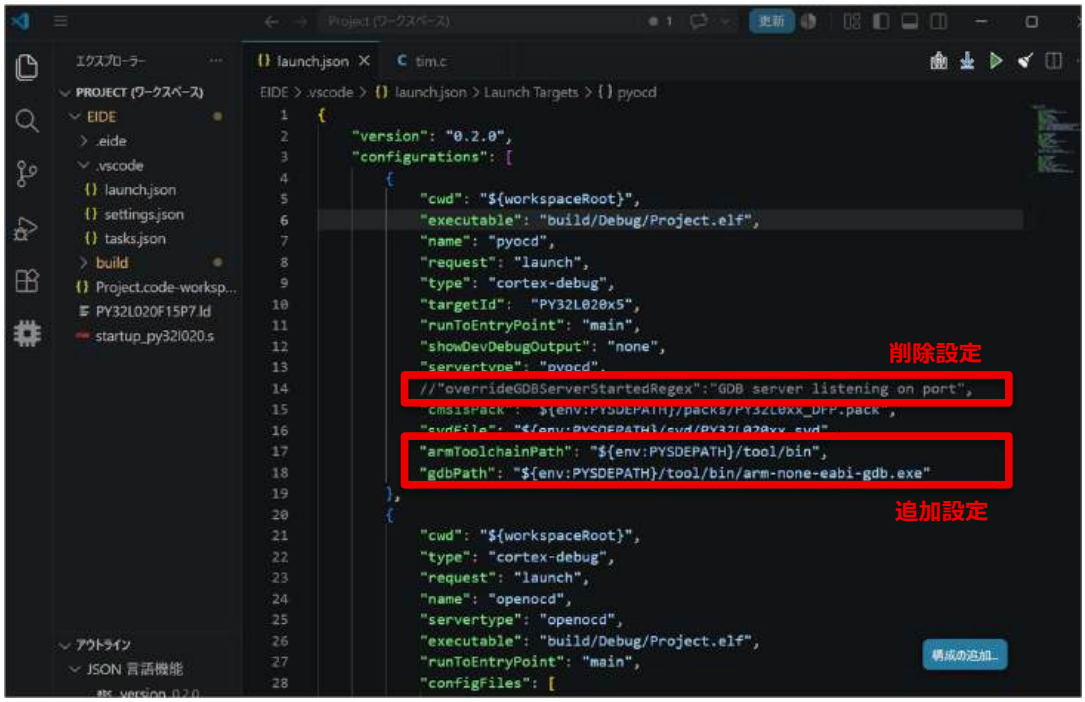
Launch.json に以下設定を追加します。

"armToolchainPath": "${env:PYSDEPATH}/tool/bin"

"gdbPath": "${env:PYSDEPATH}/tool/bin/arm-none-eabi-gdb.exe"

8.1 対処法①　Windows の環境変数に変数名と変数値を追加。

8.2 対処法②、③　launch.json の追加と削除



9. 製品情報とサポート

PY32 シリーズに関する情報は以下の各ウェブサイトから入手できます。

- PUYA セミコンダクタ（PY32 紹介）　https://www.puyasemi.com/en/py32_series.html
- Open PUYA（ドキュメント集)　https://py32.org/en/mcu/
- 佐鳥電機（問い合わせフォーム）　https://www.satori.co.jp/product/contact05.html

10. ご注意

- 本資料に記載された回路，ソフトウェアおよびこれらに関連する情報は、対象製品の使用例を説明するものです。実際の製品開発についてはお客様の責任において本資料を使用してください。これらの使用に起因して生じた損害（お客様 または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は一切その責任を負いません。
- 本資料に記載された製品データ，図，表，プログラム，アルゴリズム，応用回路例等の情報の使用に起因して発生した第三者の特許権，著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は何らの保証を行うものではなく、また責任を負うものではありません。
- 当社は、本資料に基づき当社または第三者の特許権，著作権その他の知的財産権を何ら許諾するものではありません。
- 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

11. 改訂履歴

Rev.	発行日	改訂内容
1.00	2026.08.03	新規発行