概述

PTIDE 是一款基於 Visual Studio Code 擴充功能所設計的整合開發環境(IDE)，用於支援 PTC 的 ARM 架構微處理器(MCU)。

目錄

1 簡介 4
1.1 支援的裝置 4
1.2 系統需求 4
1.2.1 作業系統 4
1.2.2 儲存空間 4
1.2.3 必要軟體 4
1.2.4 必要硬體 4
1.2.5 PTIDE 功能總覽 4
2 安裝與環境設定 5
2.1 安裝 Visual Studio Code 5
2.2 下載 PTIDE 相關安裝檔案 5
2.3 安裝 PTIDE 延伸模組 5
2.4 安裝延伸模組 6
2.5 設定工具鍊 (Toolchains) 6
2.5.1 支援的工具鍊 6
2.5.2 安裝 GNU Arm Embedded Toolchain 6
2.5.3 設定 Keil 工具鏈 (armcc / armclang) 7
2.6 設定 OpenOCD 7
3 PTIDE 介面指引 8
3.1 PTIDE 視窗介紹: PT_IDE PROJECTS 8
3.1.1 Build、Rebuild、Program Flash 和 Clean 操作按鈕 9
3.1.2 Project Resources 9
3.1.3 Chip Support Package 9
3.1.4 Builder Configurations 9
3.1.5 Flasher Configurations 10
3.1.6 Project Attributes 10
3.1.7 Project Settings 10
3.2 PTIDE 專案工作區 (workspace) 10
4 建立 PTIDE 專案 11
4.1 建立新專案 (new project) 11
4.2 建立本地範本 (local template) 12
4.3 快速開發 - 匯入本地範本 13
4.4 快速開發 - Keil 專案的匯入與轉換 13
5 程序燒錄 15
6 除錯功能 15
6.1 如何進入除錯模式 15
6.2 第一次除錯注意事項 18
6.3 監看功能 20
6.4 檢視組合語言 20
7 其他功能 21
7.1 靜態庫生成 21
8 文件版本紀錄 23

表格目錄

表 1. 文件修訂版本記錄.....23

1 簡介

1.1 支援的裝置

PT2E01 系列。

1.2 系統需求

本節說明使用 PTIDE 進行開發所需的軟硬體環境。

1.2.1 作業系統

Windows 10 / Windows 11 (64-bit)

1.2.2 儲存空間

安裝(含 PT_SDK)與開發需求至少 1 GB 可用空間

1.2.3 必要軟體

- Visual Studio Code (建議使用最新版本)
- PTIDE Extension (ptide-X.X.X.vsix)，XXX 為版本號
- Arm-Debugger 延伸模組
- GNU Arm Embedded Toolchain (gcc_arm.7z)
- OpenOCD (openocd.7z)
- Keil MDK v5 (選用)
- C/C++ 延伸模組 (選用)
- C/C++ Extension Pack 延伸模組 (選用)

1.2.4 必要硬體

- PT2E01 開發板或相容硬體，如 PTEK 開發板
- 支援 CMSIS-DAP 的調試器，如 PTEM-V1LITE 或 PTEM-V1AIR

1.2.5 PTIDE 功能總覽

- 支援 PTC ARM 微處理器
- 多種工具鏈支援，如 GNU Arm Embedded Toolchain (GCC) 、 Keil armcc (AC5) 和 Keil armclang (AC6)
- 專案建立與管理
- Keil 專案匯入與轉換
- Build, Rebuild, Debug and Program Flash

2 安裝與環境設定

2.1 安裝 Visual Studio Code

前往 [Visual Studio Code 官方網站] (https://code.visualstudio.com/) 下載並安裝最新版 Visual Studio Code。

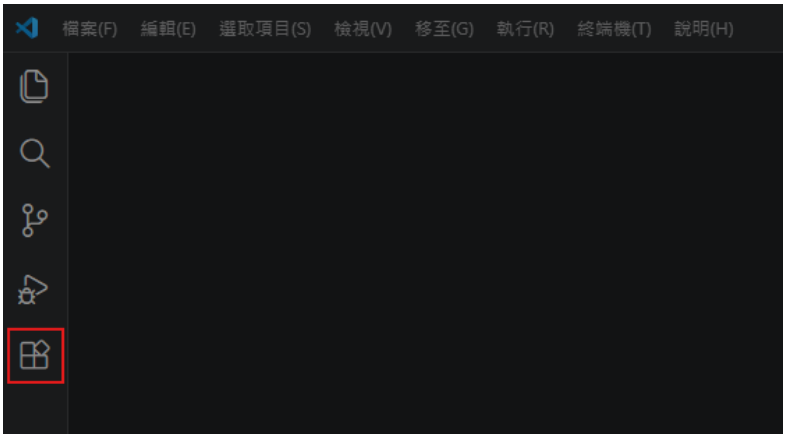
2.2 下載 PTIDE 相關安裝檔案

前往 [普誠科技官網](https://www.princeton.com.tw/) 下載 PTIDE 相關安裝檔案，包含：

- PTIDE_Extension_VX.X.X.zip (ptide-X.X.X.vsix)，XXX 為版本號
- PTIDE_Setup_VX.X.X.zip，XXX 為版本號
  - GNU Arm Embedded Toolchain (gcc_arm.7z)
  - OpenOCD (openocd.7z)

2.3 安裝 PTIDE 延伸模組

1. 開啟 VSCode
2. 點選 VSCode 左側延伸模組圖示



3. 點選延伸模組，由下拉選單中選擇 "從 VSIX 安裝..."

4. 選擇下載好的 ptide-X.X.X.vsix 檔案進行安裝 (XXX 為版本號)
5. 安裝完成 ptide-X.X.X.vsix 後 (XXX 為版本號)，若 Windows 環境中尚未安裝 dotnet-runtime-6.0.5-win-x64，系統將自動跳出安裝提示，請依指示點擊進行安裝
6. 安裝完成後，重新啟動 VSCode

2.4 安裝延伸模組

PTIDE 需要 Arm-Debugger 延伸模組來支援除錯功能，且需要 C/C++ 和 C/C++ Extension Pack 延伸模組來協助程序開發。請依照以下步驟安裝這個延伸模組：

1. 開啟 VSCode
2. 點選 VSCode 左側延伸模組圖示
3. 在搜尋欄輸入 "Arm-Debugger"，找到後點選 "安裝"
4. 同樣地，搜尋並安裝 "C/C++" 與 "C/C++ Extension Pack" 延伸模組
5. 安裝完成後，重新啟動 VSCode

2.5 設定工具鍊 (Toolchains)

本節說明如何在 PTIDE 中設定與切換編譯工具鏈。

2.5.1 支援的工具鍊

PTIDE 支援下列 Arm 工具鏈：

- GNU Arm Embedded Toolchain (GCC)
- Keil armcc (AC5)
- Keil armclang (AC6)

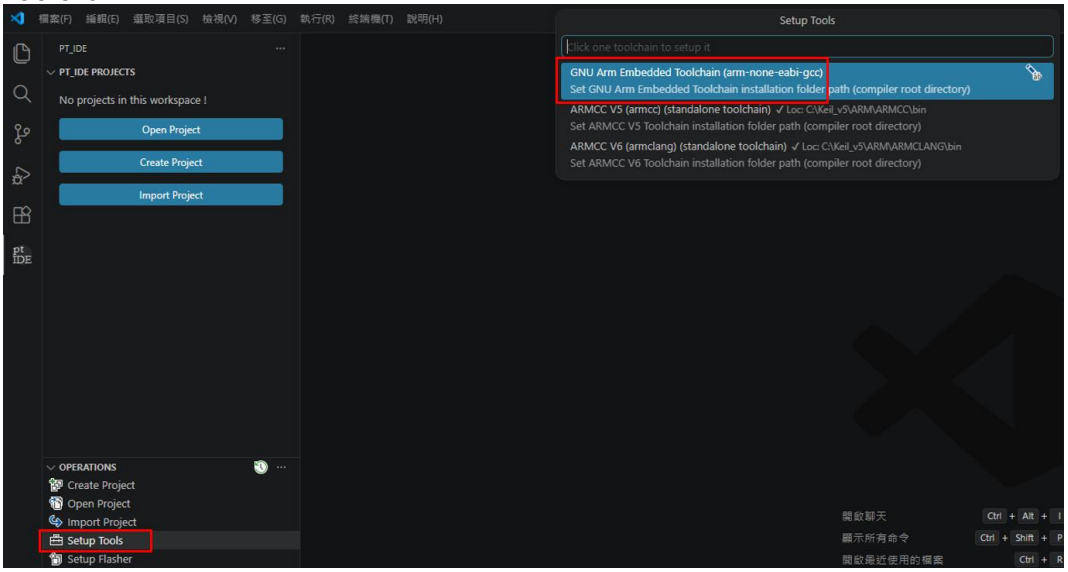
使用者可依專案需求選擇對應的工具鏈進行建置與除錯。

2.5.2 安裝 GNU Arm Embedded Toolchain

PTIDE 可透過 Setup Tools 功能完成 GNU Arm 工具鏈設定：

1. 啟動 Visual Studio Code，點選 PTIDE 圖示，進入 PTIDE 視窗

2. 在 PTIDE 視窗中，點擊 Operations 中 Setup Tools 後，選擇 GNU Arm Embedded Toolchain



3. 選擇安裝檔案(Select installation file)，找到 gcc_arm.7z

4. 確認後，PTIDE 會自動解壓並設定 GNU Arm 工具鏈路徑

完成上述步驟後，即可在專案中選擇 GCC 作為 Builder Configuration。

2.5.3 設定 Keil 工具鏈 (armcc / armclang)

若系統已安裝 Keil MDK v5，PTIDE 在首次啟動時會自動掃描並設定 armcc 與 armclang 的路徑。若偵測成功，使用者可直接在專案中選擇 AC5 或 AC6 作為編譯工具鏈。

若已安裝 Keil MDK，但仍無法使用 Keil 編譯器，以下提供 2 種解決方式:

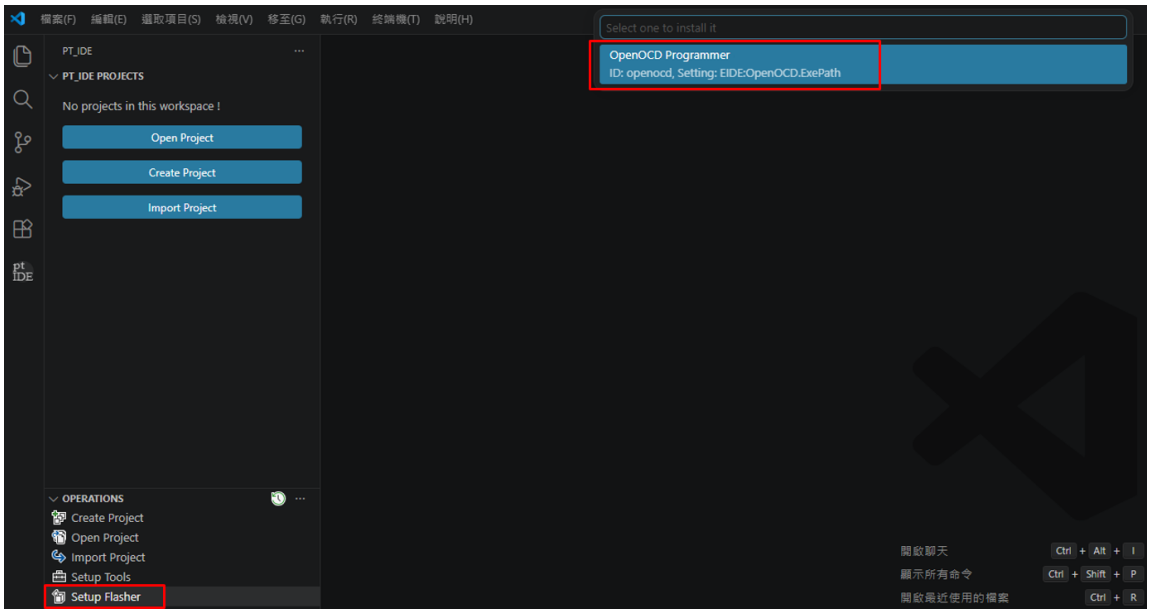
1. 點選 VSCode 左側延伸模組圖示，在搜尋欄位中搜尋 PTIDE，找到 PTIDE 擴充模組後，點擊齒輪圖示並選擇 "設定"，確認 EIDE.ARM.INI:PATH 其安裝路徑是否正確 (例如 C:\Keil_v5\UV4\UV4.exe)

2. 若確認 EIDE.ARM.INI:PATH 無誤，則可以至 Setup Tools 中手動確認 Keil 工具鏈路徑是否正確 (例如 C:\Keil_v5\ARM\ARMCC\bin)。

對於 PTIDE 而言，Keil 工具鏈為選用項目，使用者可依需求決定是否安裝與設定。

2.6 設定 OpenOCD

在 PTIDE 視窗中，點擊 Operations 中 Setup Flasher 後，選擇 OpenOCD Programmer，找到 openocd.7z 並點擊安裝。



3 PTIDE 介面指引

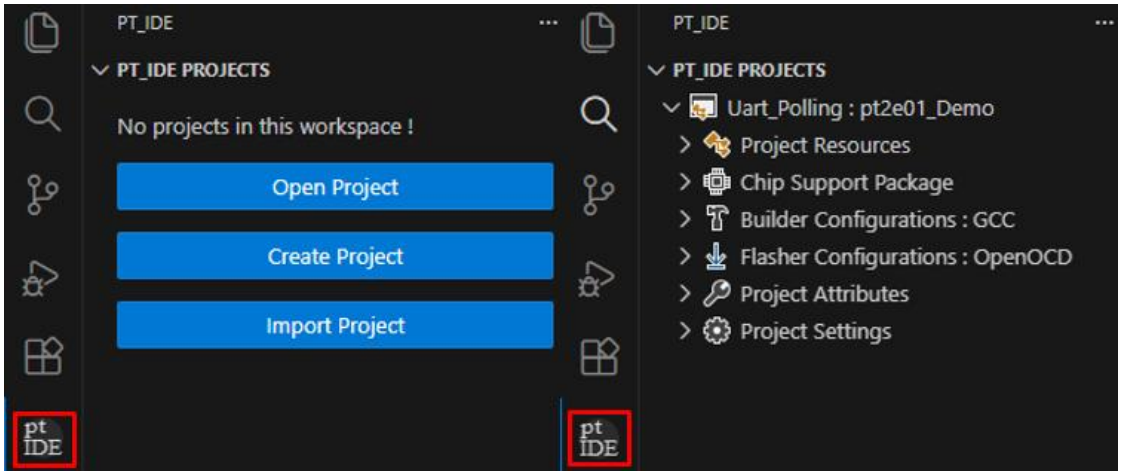
3.1 PTIDE 視窗介紹: PT_IDE PROJECTS

PTIDE 是一個基於 Visual Studio Code 的擴充功能，安裝完成後會在 VSCode 左側出現 PTIDE 圖示，點擊即可開啟 PTIDE 視窗。

◆ 下圖左半邊為 PTIDE 未開啟專案時的畫面，主要提供 3 組功能:

1. Open Project: 開啟現有 PTIDE 專案
2. Create Project: 建立新 PTIDE 專案
3. Import Project: 匯入 Keil 專案，成為 PTIDE 專案

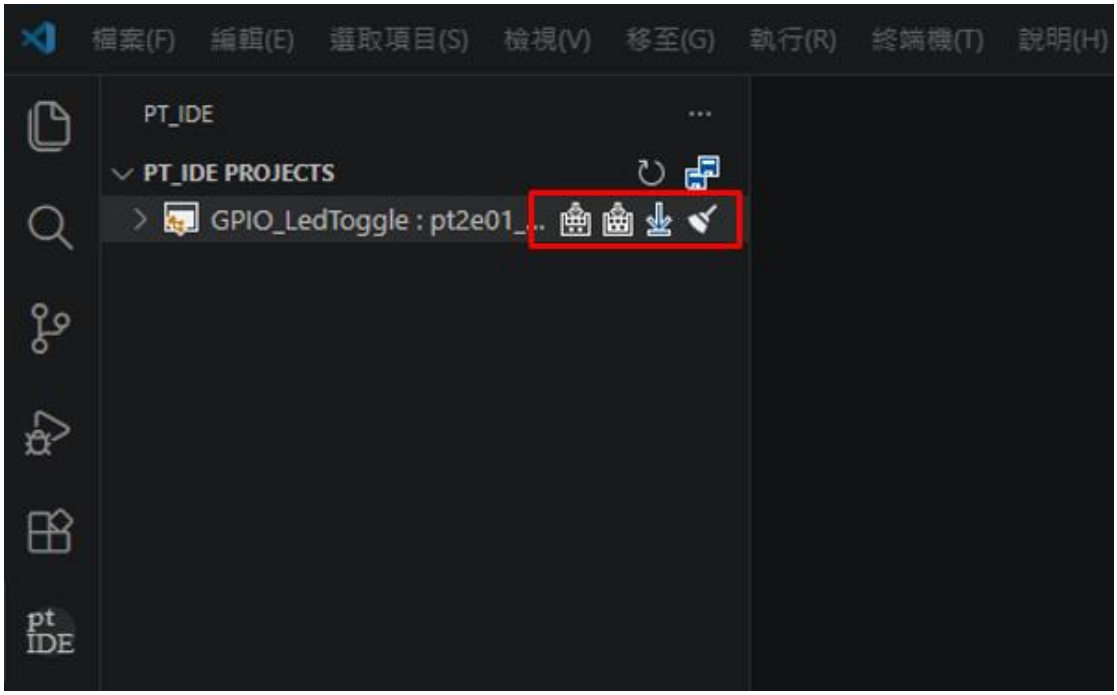
◆ 下圖右半邊圖為 PTIDE 開啟專案後的畫面，目標專案會顯示於 PT_IDE PROJECTS 區域，並提供 Build、Rebuild、Program Flash 和 Clean 等操作按鈕。後續小節將會簡介 PT_IDE PROJECTS 區域中各項功能。



3.1.1 Build、Rebuild、Program Flash 和 Clean 操作按鈕

Build、Rebuild、Program Flash 和 Clean 操作按鈕，須將滑鼠移至 PT_IDE PROJECTS 區域中專案名稱上方，才會顯示。各按鈕功能說明如下：

- Build (F7): 編譯專案，產生可執行檔
- Rebuild (Ctrl + Alt + F7): 清除先前編譯結果，重新編譯專案
- Program Flash (Ctrl + Alt + F5): 將編譯產生的可執行檔燒錄至目標裝置
- Clean: 刪除專案的編譯輸出檔案，釋放磁碟空間



3.1.2 Project Resources

Project Resources 是一個虛擬目錄，用來邏輯性地分類並顯示專案所需的檔案。此虛擬目錄僅存在於 IDE 介面中，並不一定對應實際的檔案系統目錄結構。使用者可以在這裡建立、刪除或重新命名虛擬資料夾，也可以在虛擬資料夾中新增或移除已存在的檔案。這種方式有助於組織專案檔案，使其更易於管理和瀏覽。

3.1.3 Chip Support Package

Chip Support Package 是 PTIDE 用來安裝與管理，特定晶片或系列晶片開發所需支援檔案的機制。其內容通常以 CMSIS Pack / DFP (Device Family Pack) 形式提供，包含裝置描述與開發必要檔案，讓 PTIDE 能夠識別並套用對應晶片設定。

3.1.4 Builder Configurations

Builder Configurations 是 PTIDE 用來管理專案建置 (Build) 相關設定的區域，使用者可在此選擇或切換不同的編譯工具鏈，如 GNU Arm GCC、Keil AC5/AC6，並設定連結器腳本 (Linker Script) 等建置關鍵參數。

3.1.5 Flasher Configurations

Flasher Configurations 用於設定專案的燒錄工具與相關選項。 使用者可以在此區域新增、刪除或修改不同的燒錄組態，如 erase type 和 PTEM 工具序號，以滿足不同的開發需求。

3.1.6 Project Attributes

Project Attributes 用來設定專案建置所需的路徑與參數，例如 Include Directories (標頭檔搜尋路徑)、巨集定義 (Preprocessor Defines) 和 Library Search Directories (程式庫搜尋目錄) 等。此處設定會影響編譯器在建置時如何搜尋標頭檔與解析相依檔案，若路徑設定不正確，常見會導致找不到標頭檔 (例如 xxx.h: No such file or directory) 或編譯失敗。

3.1.7 Project Settings

Project Settings 用來管理 PTIDE 專案在運行過程中所產生的檔案。使用者可以在此設定檔案的儲存位置與命名規則，以便於專案管理與版本控制。

3.2 PTIDE 專案工作區 (workspace)

PTIDE 的專案工作區 (workspace) 是一種用來組織與管理多個專案的機制，位於 VSCode 左側檔案總管中。當使用 PTIDE 開啟或建立專案時，系統會自動為該專案建立一個對應的工作區檔案 (.code-workspace)。此工作區檔案包含了專案的相關設定與配置，讓使用者能夠方便地在 VSCode 中切換與管理不同的 PTIDE 專案。使用者可以在這裡建立、刪除或重新命名實體資料夾，也可以在實體資料夾中新增或刪除檔案。

注: 建議使用 PTIDE 時，確保 PTIDE 僅開啟單一專案且與工作區中的專案是相互對應的，以避免因專案設定不一致而導致的建置或除錯問題。

4 建立 PTIDE 專案

4.1 建立新專案 (new project)

依照以下步驟建立一個全新的 PTIDE 專案:

1. 開啟 PTIDE 視窗，確認 PT_IDE PROJECTS 中無其他專案
2. 若有，請先在該專案上方點擊右鍵，選擇關閉專案
3. 點選 PT_IDE PROJECTS 或 OPERATIONS 欄位中的 Create Project
4. 在彈出的對話框中，選擇 Empty Project
5. 輸入專案名稱 (Project Name) 及選擇專案儲存路徑 (Project Location)
6. VSCode 右下方會跳出工作區轉換提示，請點擊確定
7. 此時 PT_IDE PROJECTS 區域中已有基礎專案骨架
8. 安裝 Keil Pack
   a. 請先至普誠科技官網下載所需的 Keil Pack 檔案
   b. 於 PTIDE 視窗中，點擊 PT_IDE PROJECTS 欄位中 Chip Support Package 旁的 Install CMSIS Chip Packet 按鈕，選擇下載好的 pack 檔案並安裝
   c. 點開 Chip Support Package 下拉選項，在安裝後的檔案旁點擊 Select Device 按鈕，選擇目標裝置，完成裝置設定
9. 創立並引用新檔案，例如 main.c 檔案
   a. 點擊 VSCode 左側檔案總管，到 Project 工作區新增資料夾，並在該資料夾中創立 main.c 檔案
   b. 編輯 main.c 檔案，撰寫程式碼
   c. 回到 PTIDE 視窗，於 Project Resources 中新增一虛擬資料夾，並將剛創立的 main.c 檔案加入該虛擬資料夾
10. 新增系統檔案 (system_xxx.c 與 startup_xxx.c)
    a. 於 Project Resources 中新增名稱為 "system" 的虛擬資料夾，並在該資料夾中新增 system_xxx.c 與 startup_xxx.c 檔案 (xxx 代表目標 MCU 型號)
    b. 檔案位置: .pack\Princeton\ "Keil Pack 名稱" \Device\ "目標 MCU 型號" \GCC\Source
11. 新增連結器腳本 (Linker Script)
    a. 於 Builder Configuration 底下的 Linker Script File Path 中，新增適當的連結器腳本檔案 (xxx.ld)
    b. 檔案位置: .pack\Princeton\ "Keil Pack 名稱" \Device\ "目標 MCU 型號" \GCC\Source

12. 引用已存在的檔案

    a. 於 Project Resources 中，新增所需的虛擬資料夾，並將已存在的檔案加入對應的虛擬資料夾中

    b. 於 Project Attributes 底下的 Include Directories 旁，點擊 Add Include Path 按鈕，新增系統檔案與專案所需標頭檔的搜尋路徑

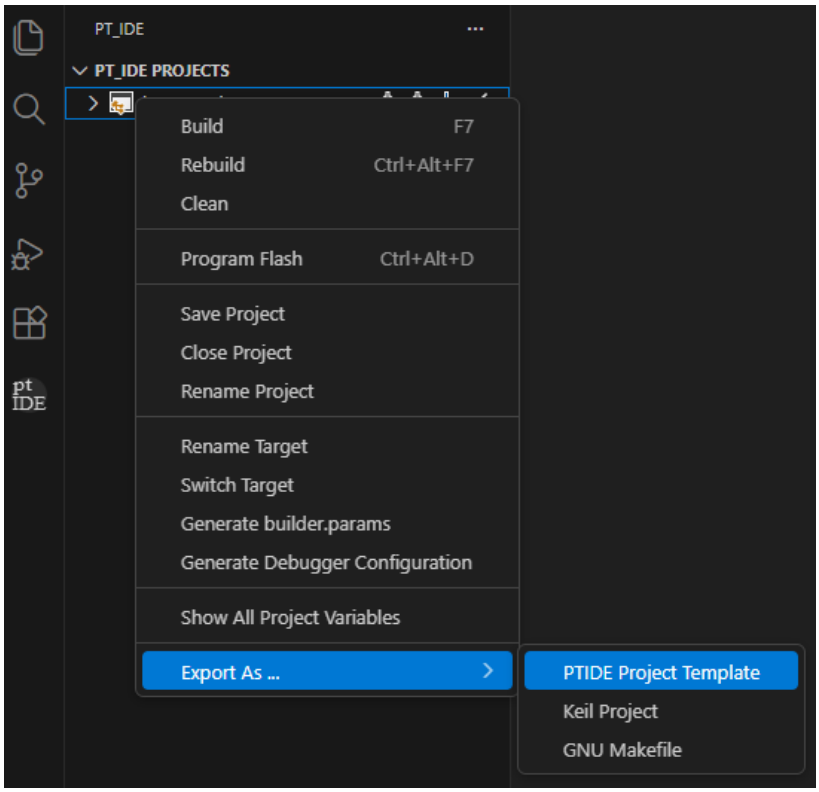
13. 此專案即完成建立，將滑鼠移到 PT_IDE PROJECTS 區域中專案名稱上方，點擊 Build (F7) / Rebuild (Ctrl + Alt + F7) 按鈕進行建置

4.2 建立本地範本 (local template)

本地範本是指已存在於使用者電腦中的專案範本，使用者可以利用這些範本快速建立新的 PTIDE 專案。以下說明如何使用本地範本建立 PTIDE 專案:

1. 開啟欲轉換為範本的現有 PTIDE 專案

2. 將滑鼠移到 PT_IDE PROJECTS 區域中專案名稱上方，右鍵點擊，選擇 Export As PTIDE Project Template



3. 即可在工作區中看到 xxx.ept 檔案，該檔案即為本地範本(xxx 代表專案名稱)

4.3 快速開發 - 匯入本地範本

依照以下步驟匯入本地範本並建立 PTIDE 專案:

1. 開啟 PTIDE 視窗，確認 PT_IDE PROJECTS 中無其他專案
2. 若有，請先在該專案上方點擊右鍵，選擇關閉專案
3. 點選 PT_IDE PROJECTS 或 OPERATIONS 欄位中的 Create Project
4. 在彈出的對話框中，選擇 Local Template，搜尋並點擊欲匯入的本地範本檔案 (xxx.ept)
5. 輸入專案名稱 (Project Name) 及選擇專案儲存路徑 (Project Location)
6. VSCode 右下方會跳出工作區轉換提示，請點擊確定
7. 此時 PT_IDE PROJECTS 區域中已建立新的專案，內容與本地範本相同

4.4 快速開發 - Keil 專案的匯入與轉換

◆ PTIDE 支援將 Keil MDK v5 專案匯入並轉換為 PTIDE 專案，依照以下步驟進行操作:

1. 開啟 PTIDE 視窗，確認 PT_IDE PROJECTS 中無其他專案
2. 若有，請先在該專案上方點擊右鍵，選擇關閉專案
3. 點選 PT_IDE PROJECTS 或 OPERATIONS 欄位中的 Import Project
4. 在彈出的對話框中，選擇 MDK arm projects
5. 選擇欲匯入的 Keil 專案檔案 (.uvprojx)
6. VSCode 右下方會跳出專案儲存位置選擇提示
   a. 選擇"是"，則匯入的專案會儲存在與 Keil 專案相同的目錄下
   b. 選擇"否"，則需指定新的專案儲存路徑，且需要自行建立專案資料夾
7. PTIDE 專案建立完成後，VSCode 右下方會跳出工作區轉換提示，請點擊確定
8. 此時 PT_IDE PROJECTS 區域中已建立新的專案，內容與匯入的 Keil 專案相同，且已可以進行 Build & Program Flash 操作
   a. Project Resources 會依 Keil 專案中的檔案結構建立虛擬目錄，虛擬目錄中的所有檔案皆與原 Keil 專案相同，在 PTIDE 修改專案中的任意檔案，均會同步更新至原 Keil 專案
   b. Builder Configuration 會自動設定為 Keil AC5
   c. Flasher Configuration 中的 Erase Type 預設為 full_erase

◆ 專案客製化設定 (接續上述流程)

1. 將原 Keil 專案中須修改的使用者檔案 (如 main.c 和 main.h) 複製到該 PTIDE 專案的儲存位置中

2. 點擊 VSCode 左側[檔案總管]，即可以在工作區中看到剛複製的檔案
3. 回到 PTIDE 視窗
   a. 於 Project Resources 中移除原 Keil 專案中須修改的使用者檔案，並將剛複製的檔案加入對應的虛擬資料夾中
   b. 於 Project Attributes 中，根據需要更新 Include Directories 以確保專案能正確編譯
4. 完成上述步驟後，可以點擊 PT_IDE PROJECTS 區域中專案名稱上方的 Build (F7) / Rebuild (Ctrl + Alt + F7) 按鈕進行建置，再點開所有須修改的使用者檔案，即可以在 PTIDE 中確認該檔案的引用路徑是否正確無誤
5. 確認無誤後，即可以開始對使用者檔案進行修改與開發
6. 若需要創立新檔案，請參考 Section 4.1, “建立新專案 (new project)” 小節中 "創立並引用新檔案" 的說明進行操作

◆ 工具鍊轉換 AC5 → GCC (接續上述流程)

1. 安裝 Keil Pack
   a. 請先至普誠科技官網(https://www.princeton.com.tw/)下載所需的 Keil Pack 檔案
   b. 於 PTIDE 視窗中，點擊 PT_IDE PROJECTS 欄位中 Chip Support Package 旁的 Install CMSIS Chip Packet 按鈕，選擇下載好的 pack 檔案並安裝
   c. 點開 Chip Support Package 下拉選項，在安裝後的檔案旁點擊 Select Device 按鈕，選擇目標裝置，完成裝置設定
2. 切換工具鍊
   a. 點擊 Builder Configuration 旁的 Select Toolchain 按鈕
   b. 在彈出的對話框中，選擇 GNU Arm Embedded Toolchain GCC
3. 新增連結器腳本 (Linker Script)
   a. 於 Builder Configuration 底下的 Linker Script File Path 中，新增適當的連結器腳本檔案 (xxx.ld)
   b. 檔案位置: .pack\Princeton\ "Keil Pack 名稱" \Device\ "目標 MCU 型號" \GCC\Source
4. 取代特定檔案，以確保專案能正確編譯
   a. 移除 Project Resources 中的 retarget.c 與 startup_xxx.s 檔案
   b. 對應新增 retarget.c 與 startup_xxx.c 檔案
   c. 檔案位置: .pack\Princeton\ "Keil Pack 名稱" \Device\ "目標 MCU 型號" \GCC\Source
5. 此時即完成工具鍊轉換，將滑鼠移到 PT_IDE PROJECTS 區域中專案名稱上方，點擊 Build

(F7) / Rebuild (Ctrl + Alt + F7) 按鈕進行建置

6. 當建置成功後，即表示完成了完整的 Keil 專案匯入與轉換流程，後續即可在 PTIDE 中進行開發與除錯

5 程序燒錄

PTIDE 支援使用 OpenOCD 進行程序燒錄，以下說明如何使用 PTIDE 進行燒錄操作:

1. 確認 PTEM 調試器已連接至電腦，並且調試器已正確連接至目標裝置
2. 點擊 Rebuild (Ctrl + Alt + F7) 按鈕，並確保該專案已成功建置，產生可執行檔 (HEX / BIN)
3. 下拉 Flasher Configuration
   a. 根據需要，點擊 Erase option 旁的 Modify 按鈕，選擇適當的擦除類型 (full_erase / sector_erase / no_erase)
   b. 點擊 Debug Probe Serial 旁的 Modify 按鈕，由下拉選單當中選擇所需的 PTEM 裝置
   c. 使用者可以根據需求，調整燒錄檔案。預設為該專案所產生的 HEX 檔案，若需更改，請點擊 Program File 旁的 Modify 按鈕，選擇其他檔案
4. 將滑鼠移到 PT_IDE PROJECTS 區域中專案名稱上方，點擊 Program Flash (Ctrl + Alt + D) 按鈕進行燒錄操作
5. 燒錄過程中，請注意 PTIDE 視窗下方的輸出視窗，以確認燒錄是否成功完成

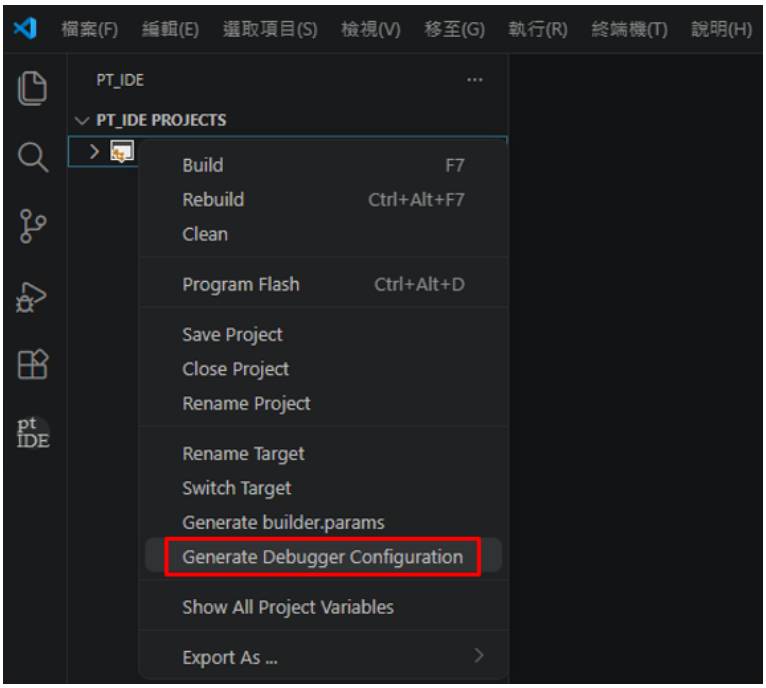
6 除錯功能

6.1 如何進入除錯模式

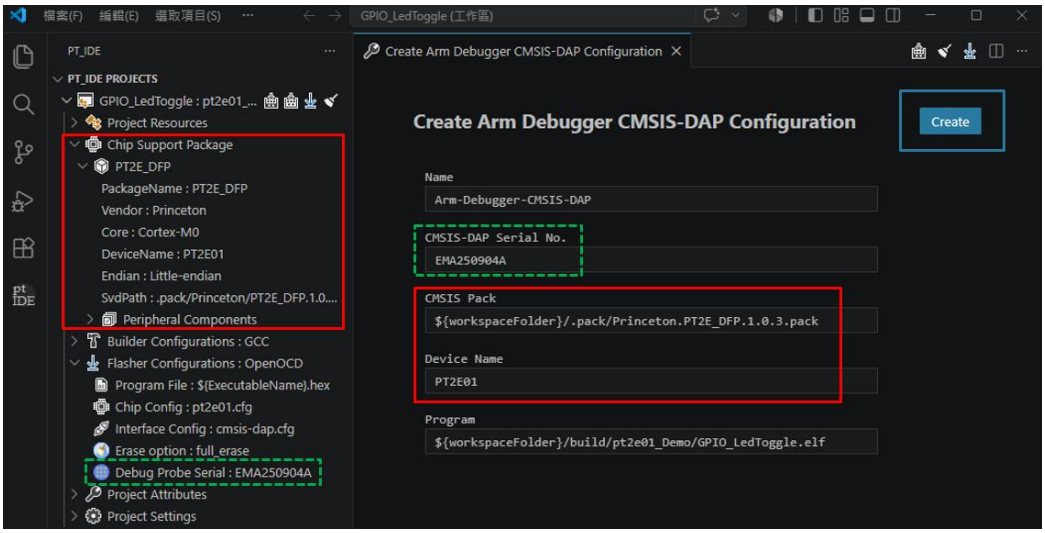
PTIDE 支援使用 Arm-Debugger 延伸模組進行除錯操作，以下說明如何使用 PTIDE 進行除錯:

1. 確認 PTEM 調試器已連接至電腦，並且調試器已正確連接至目標裝置
2. 打開欲除錯的 PTIDE 專案
3. 確認 Flasher Configuration 中的 Debug Probe Serial 已選擇正確的 PTEM 裝置
4. 生成 launch.json 檔案
   a. 生成 launch.json 檔案前，請先確保此專案已安裝 Keil Pack，並選擇正確的目標裝置。安裝方式請參考 Section 4.1, “建立新專案 (new project)” 小節中 "安裝 Keil Pack" 的說明進行操作
   b. 將滑鼠移到 PT_IDE PROJECTS 區域中專案名稱上方，右鍵點擊，選擇 Generate

Debugger Configuration

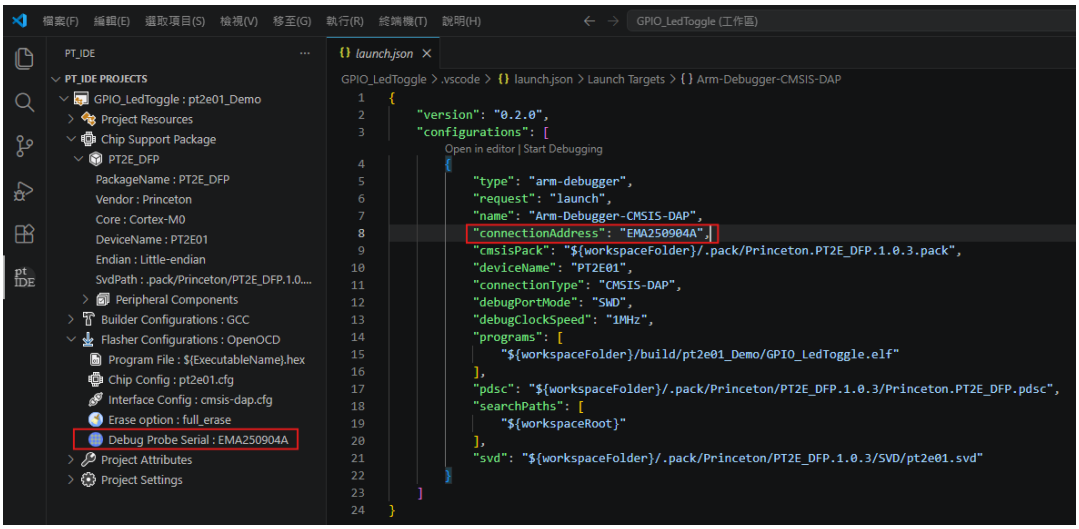


c. 顯示 Create Arm Debugger CMSIS-DAP Debug Configuration 視窗，依據需求修改調試器序號後，點擊 Create



d. 最終生成 launch.json 檔案

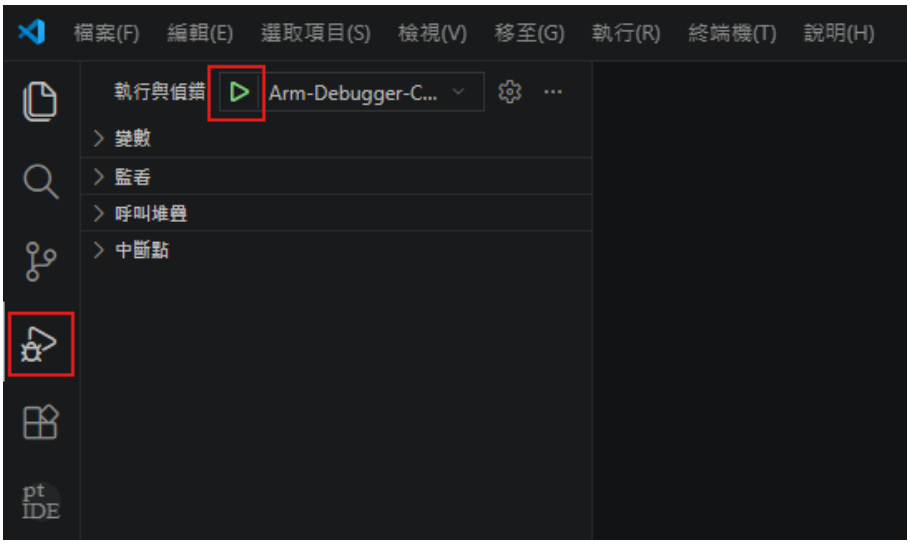
5. 若專案已生成 launch.json 檔案，須注意調試器序號是否正確



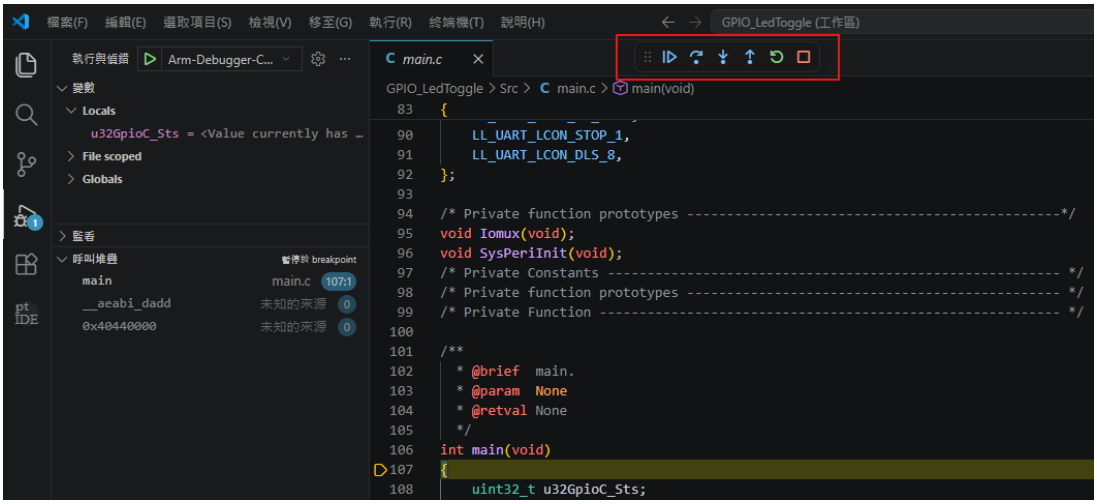
6. 開始除錯前，請先點擊 Rebuild (Ctrl + Alt + F7) 按鈕，並確保該專案已成功建置，再點擊 Program Flash (Ctrl + Alt + D) 按鈕，將可執行檔燒錄至目標裝置

7. 點擊 VSCode 左側除錯圖示，進入除錯視窗

8. 在[除錯視窗]中，點擊綠色按鈕 (Start Debugging) 或按下 F5 鍵，開始除錯程序。若為第一次除錯，可以參考 Section 6.2, "第一次除錯注意事項" 小節相關內容



9. 看到[除錯工具列](如紅框中所示)，表示成功進入除錯模式，工具列由左至右分別為:



a. 繼續 / 暫停 (Continue / Pause): 繼續執行程式直到下一個斷點 / 暫停執行程序

b. 不進函式 (Step Over): 執行當前行並停在下一行

c. 逐步執行 (Step Into): 進入函式內部並停在第一行

d. 跳離函式 (Step Out): 執行完當前函式並停在該函式的下一行

e. 重新啟動 (Restart): 停止並重新啟動除錯程序

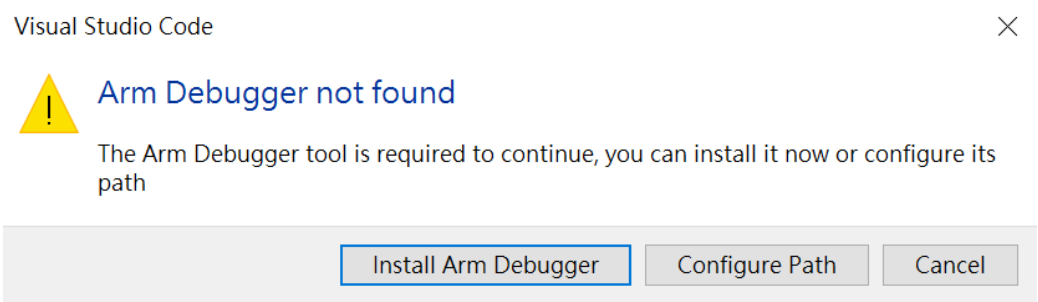
f. 停止 (Stop): 停止除錯並結束除錯模式

PTEM 裝置	序號
PTEM-V1LITE FW_V_2.2.1 之前	00110022
PTEM-V1AIR FW_V_1.0.3 之前	00110011
PTEM-V1LITE FW_V_2.2.2 之後	EML250804A or EML260601A
PTEM-V1AIR FW_V_1.0.3 之後	EMA250904A or EMK251114A
PTEM-V1	EMX260401A

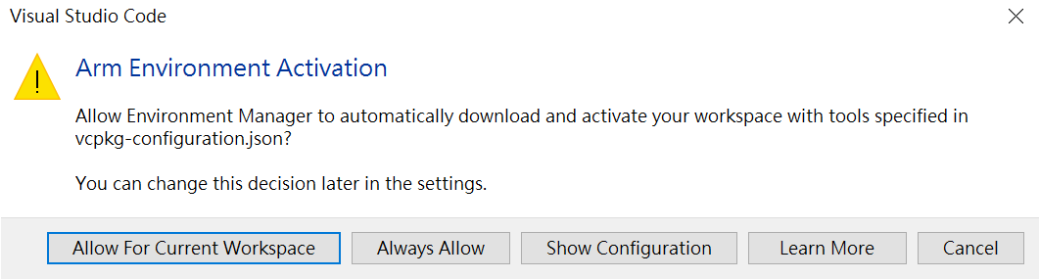
6.2 第一次除錯注意事項

本章節紀錄第一次在 PTIDE 中進入除錯模式，VSCode 可能會跳出的提示:

1. 提示安裝 Arm Debugger，請點擊安裝

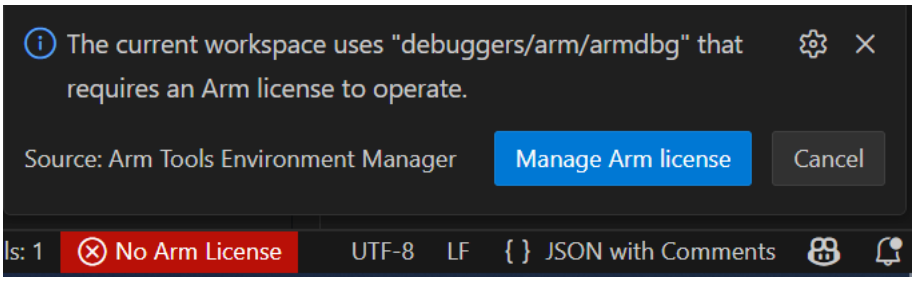


2. 提示安裝必要的除錯元件，請點擊允許

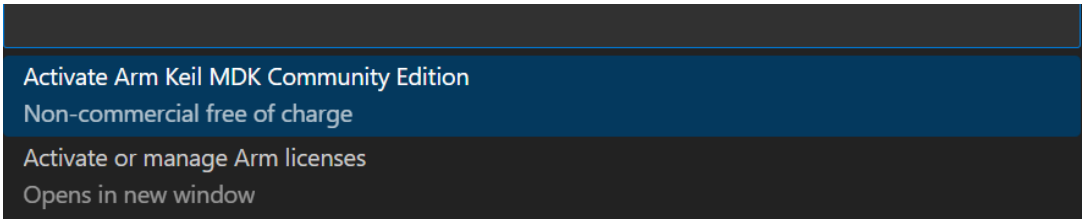


3. 提示需安裝 Arm License

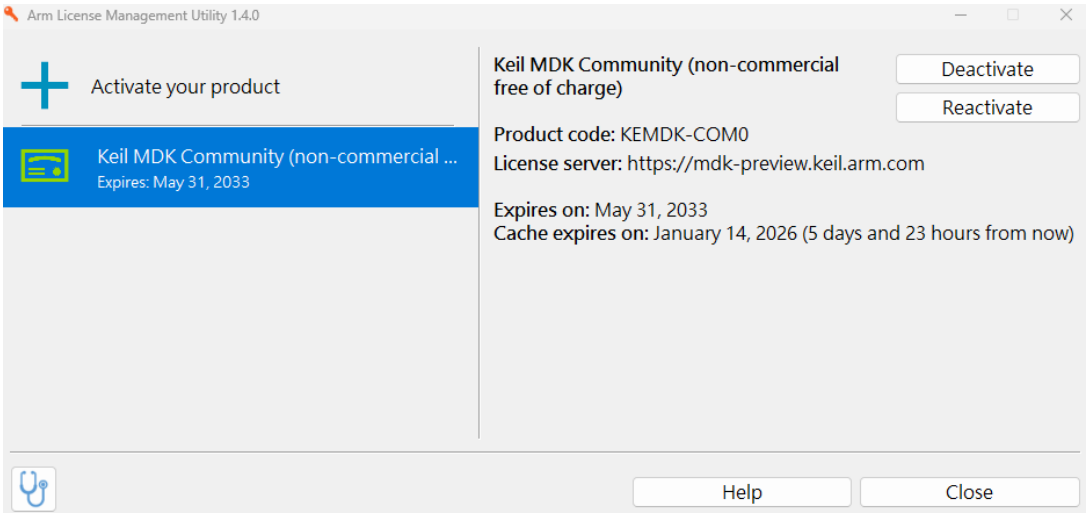
a. 點擊 Manage Arm License 按鈕或紅底 No Arm License 警示



b. 下拉選單中點擊 Activate Arm Keil MDK Community Edition



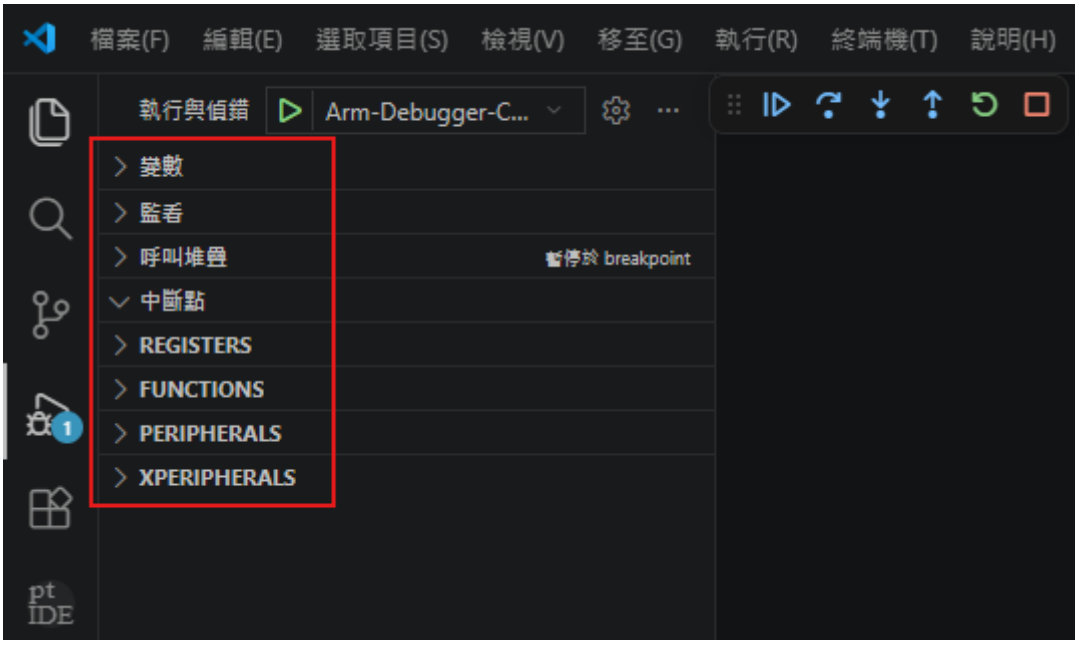
c. 在彈出視窗中點擊 Keil MDK Community，再點擊 Reactivate 按鈕



6.3 監看功能

進入 debug mode 後，使用者可以透過 VSCode 左側監看視窗，了解除錯過程中程序狀態變化。以下介紹各個視窗的功能:

- 變數: 顯示目前作用域內的所有變數及其值，使用者可以展開物件或陣列以查看其內部結構
- 監看: 使用者可以手動新增變數或表達式，以便在除錯過程中持續監控其值的變化
- 呼叫堆疊: 顯示目前執行緒的呼叫堆疊，使用者可以點擊堆疊中的任意一層，以查看該層的程式碼位置
- 中斷點: 列出所有已設定的中斷點，使用者可以刪除中斷點，或輸入函式名稱以此設定中斷點
- REGISTERS: 顯示目標裝置的 CPU 暫存器值，使用者可以查看目前暫存器的內容，以便進行低階除錯
- FUNCTIONS: 顯示目前作用域內的所有函式，使用者可以點擊前往查看該程式碼位置和設定該函式為中斷點
- PERIPHERALS / XPERIPHERALS: 顯示目標裝置的外設寄存器，使用者可以查看並修改寄存器數值，以便進行低階除錯



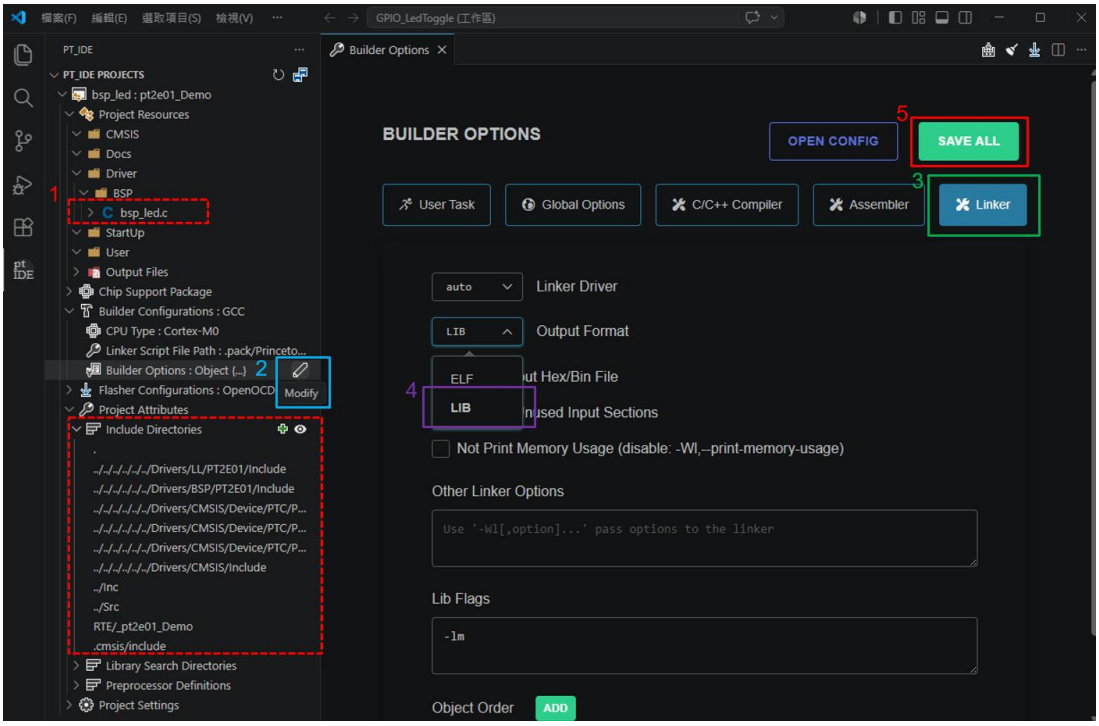
6.4 檢視組合語言

在每個 C 來源檔中點擊右鍵，選擇 "Show Disassembly" 選項，即可檢視對應的組合語言程式碼。此功能在一般模式下即可使用，無需進入除錯模式。

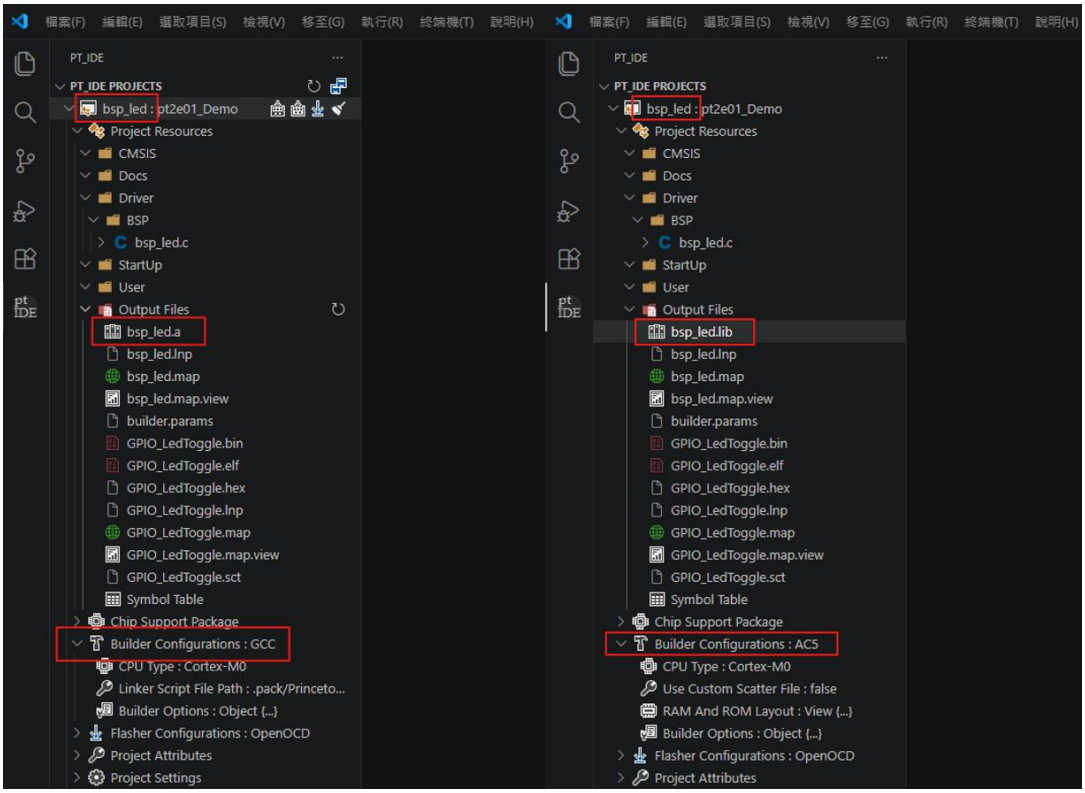
7 其他功能

7.1 靜態庫生成

1. 建構一個靜態庫專案
   - 僅包含目標(.c)檔案
   - 該目標(.c)檔案所需的所有標頭檔(.h)皆需要載入至 Project Attributes 的 Include Directories 中
2. 下拉 Builder Configuration，點擊 Builder Options 旁的 Modify 按鈕
3. 點擊 Builder Options 視窗中的 Linker 選項
4. 下拉 Output Format，選擇 LIB
5. 點擊 SAVE ALL



6. 此專案即完成建立，將滑鼠移到 PT_IDE PROJECTS 區域中專案名稱上方，點擊 Build (F7) / Rebuild (Ctrl + Alt + F7) 按鈕進行建置
7. 成功後即可以在 Output Files 看到對應的檔案(GCC: .a / AC5: .lib / AC6: .lib)，該檔案即為靜態庫檔案，預設名稱為此專案名稱，使用者可依需求修改名稱



8. 此靜態庫檔案即可以取代其他專案中的目標(.c)檔案

8 文件版本紀錄

表 1. 文件修訂版本記錄

日期	版本	修訂內容
2026-07-15	V1.0	初版