概述

PTIDE 是一款基于 Visual Studio Code 扩充功能所设计的整合开发环境(IDE)，用于支持 PTC 的 ARM 架构微处理器(MCU)。

目录

1 简介 4
1.1 支持的设备 4
1.2 系统需求 4
1.2.1 操作系统 4
1.2.2 储存空间 4
1.2.3 必要软件 4
1.2.4 必要硬件 4
1.2.5 PTIDE 功能总览 4
2 安装与环境设置 5
2.1 安装 Visual Studio Code 5
2.2 下载 PTIDE 相关安装文件 5
2.3 安装 PTIDE 延伸模块 5
2.4 安装延伸模块 6
2.5 设置工具链 (Toolchains) 6
2.5.1 支持的工具链 6
2.5.2 安装 GNU Arm Embedded Toolchain 6
2.5.3 设置 Keil 工具链 (armcc / armclang) 7
2.6 设置 OpenOCD 7
3 PTIDE 界面指引 8
3.1 PTIDE 窗口介绍: PT_IDE PROJECTS 8
3.1.1 Build、Rebuild、Program Flash 和 Clean 操作按钮 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 项目工作区 (workspace) 10
4 建立 PTIDE 项目 11
4.1 建立新项目 (new project) 11
4.2 建立本地模板 (local template) 12
4.3 快速开发 - 导入本地模板 13
4.4 快速开发 - Keil 项目的导入与转换 13
5 程序烧录 15
6 调试功能 15
6.1 如何进入调试模式 15
6.2 第一次调试注意事项 18
6.3 监看功能 20
6.4 检视汇编语言 20
7 其他功能 21
7.1 静态库生成 21
8 文件版本纪录 23

表格目录

表 1. 文件修订版本记录 23

1 简介

1.1 支持的设备

PT2E01 系列。

1.2 系统需求

本节说明使用 PTIDE 进行开发所需的软硬件环境。

1.2.1 操作系统

Windows 10 / Windows 11 (64-bit)

1.2.2 储存空间

安装(含 PT_SDK)与开发需求至少 1 GB 可用空间

1.2.3 必要软件

- Visual Studio Code (建议使用最新版本)
- PTIDE Extension (ptide-X.X.X.vsix)，XXX 为版本号
- Arm-Debugger 延伸模块
- GNU Arm Embedded Toolchain (gcc_arm.7z)
- OpenOCD (openocd.7z)
- Keil MDK v5 (选用)
- C/C++ 延伸模块 (选用)
- C/C++ Extension Pack 延伸模块 (选用)

1.2.4 必要硬件

- PT2E01 开发板或兼容硬件，如 PTEK 开发板
- 支持 CMSIS-DAP 的调试器，如 PTEM-V1LITE 或 PTEM-V1AIR

1.2.5 PTIDE 功能总览

- 支持 PTC ARM 微处理器
- 多种工具链支持，如 GNU Arm Embedded Toolchain (GCC) 、 Keil armcc (AC5) 和 Keil armclang (AC6)
- 项目建立与管理
- Keil 项目导入与转换
- Build, Rebuild, Debug and Program Flash

2 安装与环境设置

2.1 安装 Visual Studio Code

前往 [Visual Studio Code 官方网站] (https://code.visualstudio.com/) 下载并安装最新版 Visual Studio Code。

2.2 下载 PTIDE 相关安装文件

前往 [普诚科技官网](https://www.princeton.com.tw/) 下载 PTIDE 相关安装文件，包含：

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

2.3 安装 PTIDE 延伸模块

1. 打开 VSCode
2. 点击 VSCode 左侧延伸模块图示

3. 点击延伸模块，由下拉菜单中选择 "从 VSIX 安装..."

4. 选择下载好的 ptide-X.X.X.vsix 文件进行安装（XXX 为版本号）
5. 安装完成 ptide-X.X.X.vsix 后（XXX 为版本号），若 Windows 环境中尚未安装 dotnet-runtime-6.0.5-win-x64，系统将自动跳出安装提示，请依指示点击进行安装
6. 安装完成后，重新启动 VSCode

2.4 安装延伸模块

PTIDE 需要 Arm-Debugger 延伸模块来支持调试功能，且需要 C/C++ 和 C/C++ Extension Pack 延伸模块来协助程序开发。请依照以下步骤安装这个延伸模块：

1. 打开 VSCode
2. 点击 VSCode 左侧延伸模块图示
3. 在搜索栏输入 "Arm-Debugger"，找到后点击 "安装"
4. 同样地，搜索并安装 "C/C++" 与 "C/C++ Extension Pack" 延伸模块
5. 安装完成后，重新启动 VSCode

2.5 设置工具链（Toolchains）

本节说明如何在 PTIDE 中设置与切换编译工具链。

2.5.1 支持的工具链

PTIDE 支持下列 Arm 工具链：

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

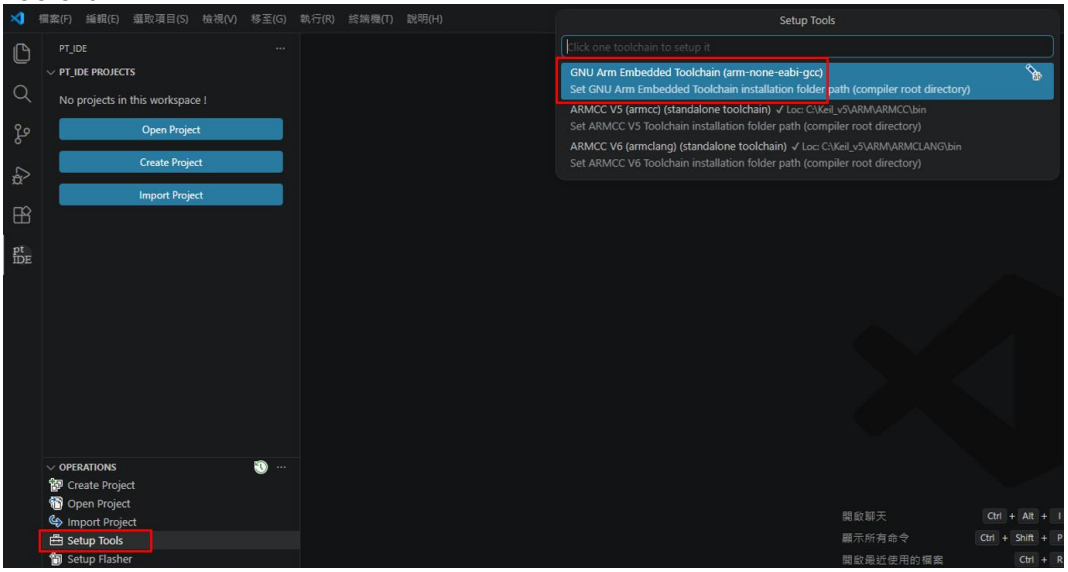
用户可依项目需求选择对应的工具链进行构建与调试。

2.5.2 安装 GNU Arm Embedded Toolchain

PTIDE 可通过 Setup Tools 功能完成 GNU Arm 工具链设置：

1. 启动 Visual Studio Code，点击 PTIDE 图标，进入 PTIDE 窗口

2. 在 PTIDE 窗口中，点击 Operations 中 Setup Tools 后，选择 GNU Arm Embedded Toolchain



3. 选择安装文件(Select installation file)，找到 gcc_arm.7z

4. 确认后，PTIDE 会自动解压并设置 GNU Arm 工具链路径

完成上述步骤后，即可在项目中选择 GCC 作为 Builder Configuration。

2.5.3 设置 Keil 工具链（armcc / armclang)

若系统已安装 Keil MDK v5，PTIDE 在首次启动时会自动扫描并设置 armcc 与 armclang 的路径。若检测成功，用户可直接在项目中选择 AC5 或 AC6 作为编译工具链。

若已安装 Keil MDK，但仍无法使用 Keil 编译程序，以下提供 2 种解决方式:

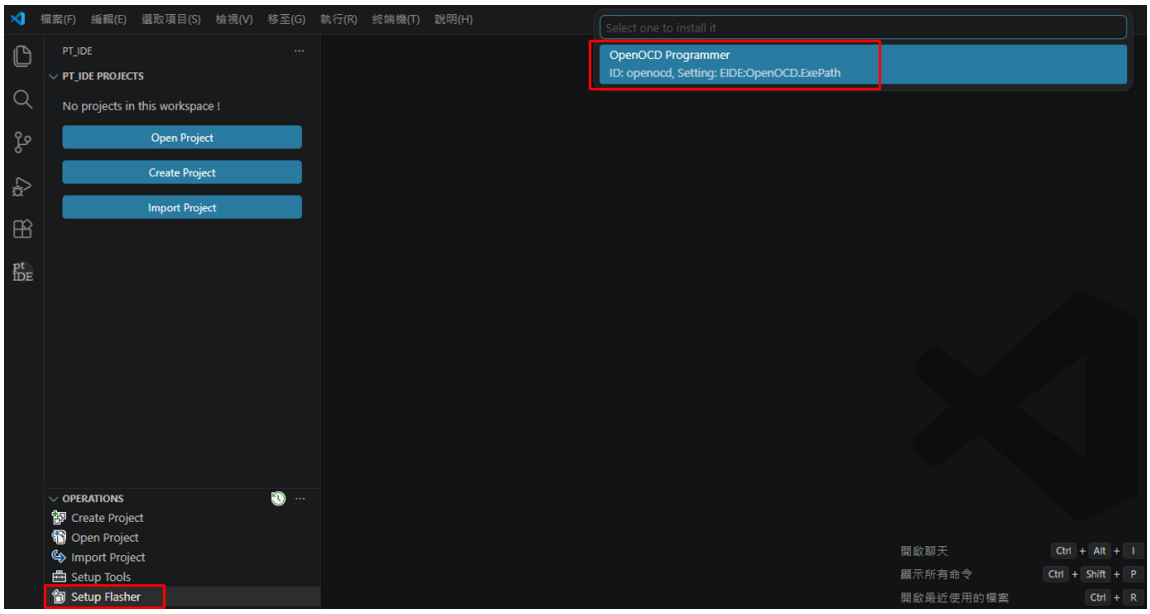
1. 点击 VSCode 左侧延伸模块图示，在搜索栏位中搜索 PTIDE，找到 PTIDE 扩充模块后，点击齿轮图示并选择 "设置"，确认 EIDE.ARM.INI:PATH 其安装路径是否正确（例如 C:\Keil_v5\UV4\UV4.exe)

2. 若确认 EIDE.ARM.INI:PATH 无误，则可以至 Setup Tools 中手动确认 Keil 工具链路径是否正确（例如 C:\Keil_v5\ARM\ARMCC\bin)。

对于 PTIDE 而言，Keil 工具链为选用项目，用户可依需求决定是否安装与设置。

2.6 设置 OpenOCD

在 PTIDE 窗口中，点击 Operations 中 Setup Flasher 后，选择 OpenOCD Programmer，找到 openocd.7z 并点击安装。



3 PTIDE 界面指引

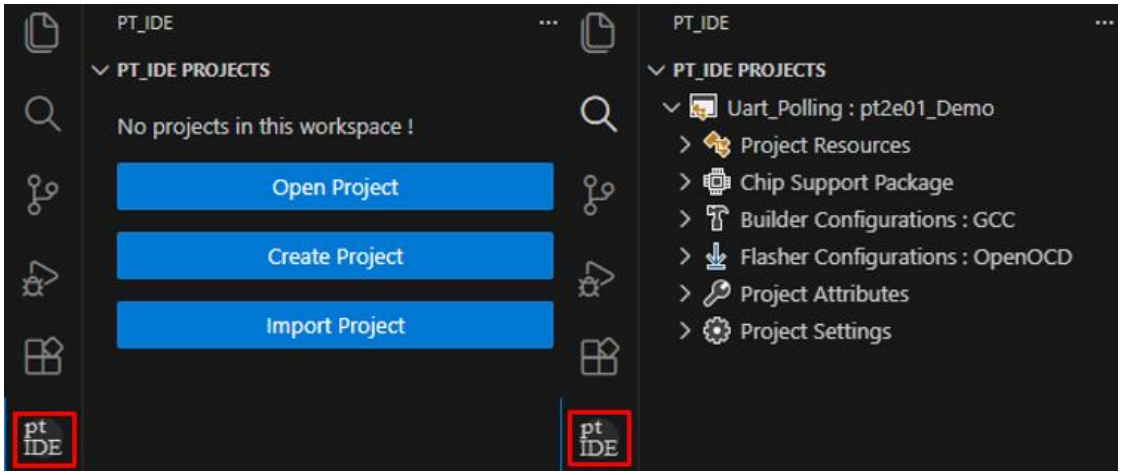
3.1 PTIDE 窗口介绍: PT_IDE PROJECTS

PTIDE 是一个基于 Visual Studio Code 的扩充功能，安装完成后会在 VSCode 左侧出现 PTIDE 图示，点击即可开启 PTIDE 窗口。

◆ 下图左半边为 PTIDE 未开启项目时的画面，主要提供 3 组功能:

1. Open Project: 开启现有 PTIDE 项目
2. Create Project: 建立新 PTIDE 项目
3. Import Project: 导入 Keil 项目，生成 PTIDE 项目

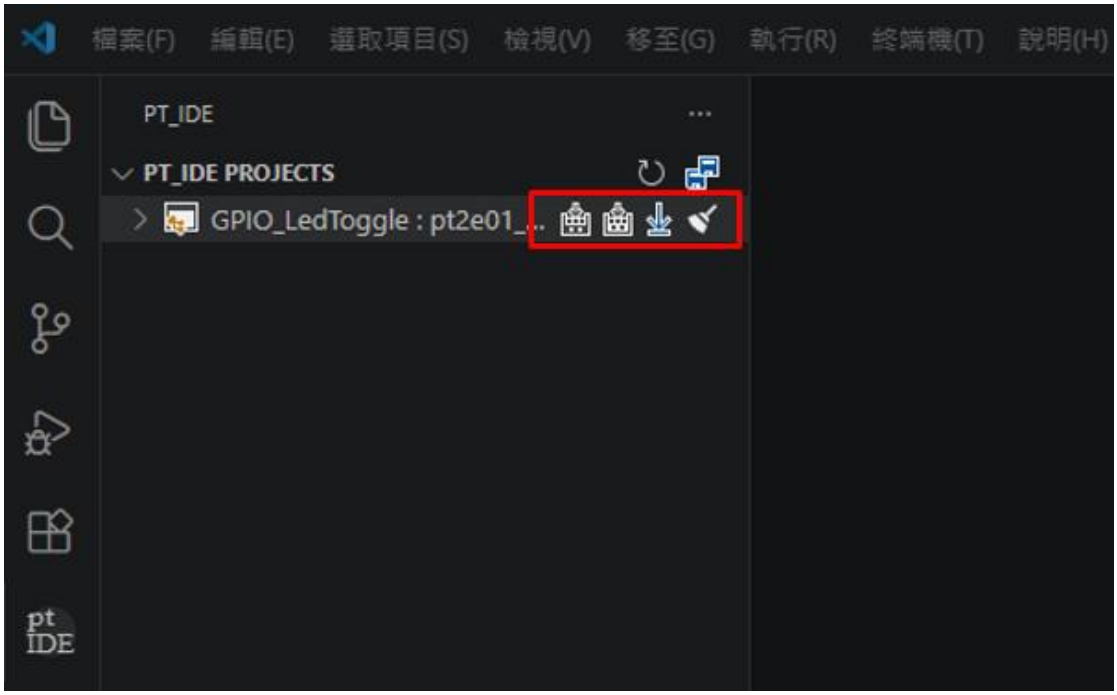
◆ 下图右半边图为 PTIDE 开启项目后的画面，目标项目会显示于 PT_IDE PROJECTS 区域，并提供 Build、Rebuild、Program Flash 和 Clean 等操作按钮。后续小节将会简要介绍 PT_IDE PROJECTS 区域中各项功能。



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

Build、Rebuild、Program Flash 和 Clean 操作按钮，须将鼠标移至 PT_IDE PROJECTS 区域中项目名称上方，才会显示。各按钮功能说明如下：

- Build (F7): 编译项目，产生可执行文件
- Rebuild (Ctrl + Alt + F7): 清除先前编译结果，重新编译项目
- Program Flash (Ctrl + Alt + F5): 将编译产生的可执行文件烧录至目标设备
- Clean: 删除项目的编译输出文件，释放磁盘空间



3.1.2 Project Resources

Project Resources 是一个虚拟目录，用来逻辑性地分类并显示项目所需的文件。此虚拟目录仅存在于 IDE 接口中，并不一定对应实际的文件系统目录结构。用户可以在这里建立、删除或重新命名虚拟文件夹，也可以在虚拟文件夹中新增或移除已存在的文件。这种方式有助于组织项目文件，使其更易于管理和浏览。

3.1.3 Chip Support Package

Chip Support Package 是 PTIDE 用来安装与管理，特定芯片或系列芯片开发所需支持文件的机制。其内容通常以 CMSIS Pack / DFP (Device Family Pack) 形式提供，包含设备描述与开发必要文件，让 PTIDE 能够识别并套用对应芯片设置。

3.1.4 Builder Configurations

Builder Configurations 是 PTIDE 用来管理项目构建 (Build) 相关设置的区域，用户可在此选择或切换不同的编译工具链，如 GNU Arm GCC、Keil AC5/AC6，并设置链接器脚本 (Linker Script) 等构建关键参数。

3.1.5 Flasher Configurations

Flasher Configurations 用于设置项目的烧录工具与相关选项。 用户可以在此区域新增、删除或修改不同的烧录配置，如 erase type 和 PTEM 工具序号，以满足不同的开发需求。

3.1.6 Project Attributes

Project Attributes 用来设置项目构建所需的路径与参数，例如 Include Directories (头文件搜索路径)、宏定义 (Preprocessor Defines) 和 Library Search Directories (链接库搜索目录) 等。此处设置会影响编译程序在构建时如何搜索头文件与解析依赖文件，若路径设置不正确，常见会导致找不到头文件 (例如 xxx.h: No such file or directory) 或编译失败。

3.1.7 Project Settings

Project Settings 用来管理 PTIDE 项目在运行过程中所产生的文件。用户可以在此配置文件的储存位置与命名规则，以便于项目管理与版本控制。

3.2 PTIDE 项目工作区 (workspace)

PTIDE 的项目工作区 (workspace) 是一种用来组织与管理多个项目的机制，位于 VSCode 左侧资源管理器中。当使用 PTIDE 开启或建立项目时，系统会自动为该项目建立一个对应的工作区文件 (.code-workspace)。此工作区文件包含了项目的相关设置与配置，让用户能够方便地在 VSCode 中切换与管理不同的 PTIDE 项目。用户可以在这里建立、删除或重新命名实体文件夹，也可以在实体文件夹中新增或删除文件。

注: 建议使用 PTIDE 时，确保 PTIDE 仅开启单一项目且与工作区中的项目是相互对应的，以避免因项目设置不一致而导致的构建或调试问题。

4 建立 PTIDE 项目

4.1 建立新项目 (new project)

依照以下步骤建立一个全新的 PTIDE 项目:

1. 开启 PTIDE 窗口，确认 PT_IDE PROJECTS 中无其他项目
2. 若有，请先在该项目上方点击右键，选择关闭项目
3. 点击 PT_IDE PROJECTS 或 OPERATIONS 栏位中的 Create Project
4. 在弹出的对话框中，选择 Empty Project
5. 输入项目名称 (Project Name) 及选择项目储存路径 (Project Location)
6. VSCode 右下方会跳出工作区转换提示，请点击确定
7. 此时 PT_IDE PROJECTS 区域中已有基础项目骨架
8. 安装 Keil Pack
   a. 请先至普诚科技官网下载所需的 Keil Pack 文件
   b. 于 PTIDE 窗口中，点击 PT_IDE PROJECTS 栏位中 Chip Support Package 旁的 Install CMSIS Chip Packet 按钮，选择下载好的 pack 文件并安装
   c. 点开 Chip Support Package 下拉选项，在安装后的文件旁点击 Select Device 按钮，选择目标设备，完成设备设置
9. 创立并引用新文件，例如 main.c 文件
   a. 点击 VSCode 左侧资源管理器，到 Project 工作区新增文件夹，并在该文件夹中创立 main.c 文件
   b. 编辑 main.c 文件，撰写代码
   c. 回到 PTIDE 窗口，于 Project Resources 中新增一虚拟文件夹，并将刚创立的 main.c 文件加入该虚拟文件夹
10. 新增系统文件 (system_xxx.c 与 startup_xxx.c)
    a. 于 Project Resources 中新增名称为 "system" 的虚拟文件夹，并在该文件夹中新增 system_xxx.c 与 startup_xxx.c 文件 (xxx 代表目标 MCU 型号)
    b. 文件位置: .pack\Princeton\ "Keil Pack 名称" \Device\ "目标 MCU 型号" \GCC\Source
11. 新增链接器脚本 (Linker Script)
    a. 于 Builder Configuration 底下的 Linker Script File Path 中，新增适当的链接器脚本文件 (xxx.ld)
    b. 文件位置: .pack\Princeton\ "Keil Pack 名称" \Device\ "目标 MCU 型号" \GCC\Source

12. 引用已存在的文件

    a. 于 Project Resources 中，新增所需的虚拟文件夹，并将已存在的文件加入对应的虚拟文件夹中

    b. 于 Project Attributes 底下的 Include Directories 旁，点击 Add Include Path 按钮，新增系统文件与项目所需头文件的搜索路径

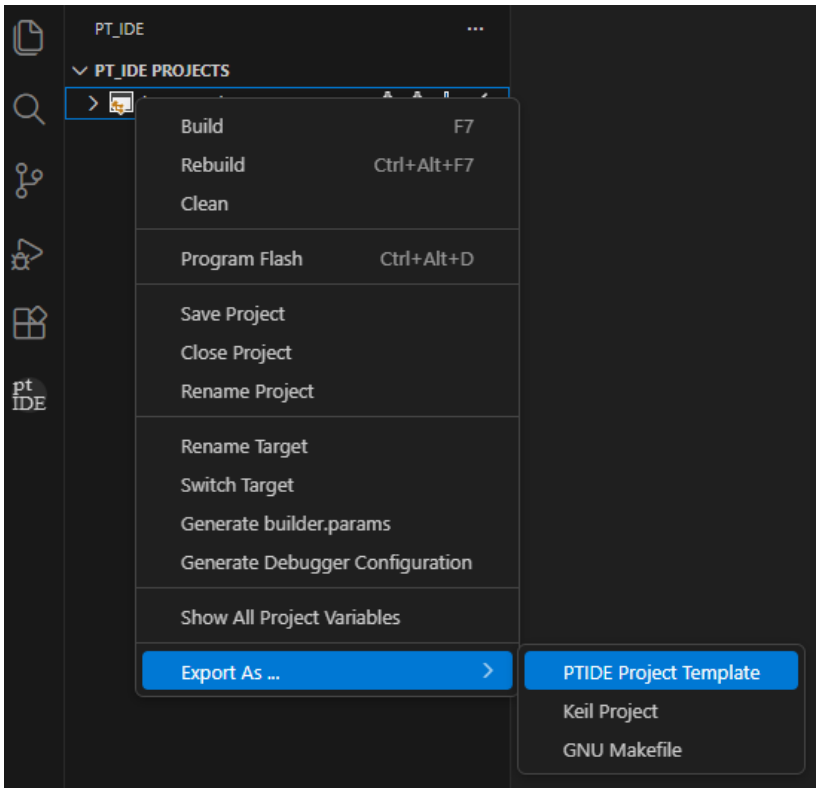
13. 此项目即完成建立，将鼠标移到 PT_IDE PROJECTS 区域中项目名称上方，点击 Build (F7) / Rebuild (Ctrl + Alt + F7) 按钮进行构建

4.2 建立本地模板 (local template)

本地模板是指已存在于用户计算机中的项目模板，用户可以利用这些模板快速建立新的 PTIDE 项目。以下说明如何使用本地模板建立 PTIDE 项目：

1. 开启欲转换为模板的现有 PTIDE 项目

2. 将鼠标移到 PT_IDE PROJECTS 区域中项目名称上方，右键点击，选择 Export As PTIDE Project Template



3. 即可在工作区中看到 xxx.ept 文件，该文件即为本地模板(xxx 代表项目名称)

4.3 快速开发 - 导入本地模板

依照以下步骤导入本地模板并建立 PTIDE 项目:

1. 开启 PTIDE 窗口，确认 PT_IDE PROJECTS 中无其他项目
2. 若有，请先在该项目上方点击右键，选择关闭项目
3. 点击 PT_IDE PROJECTS 或 OPERATIONS 栏位中的 Create Project
4. 在弹出的对话框中，选择 Local Template，搜索并点击欲导入的本地模板文件 (xxx.ept)
5. 输入项目名称 (Project Name) 及选择项目储存路径 (Project Location)
6. VSCode 右下方会跳出工作区转换提示，请点击确定
7. 此时 PT_IDE PROJECTS 区域中已建立新的项目，内容与本地模板相同

4.4 快速开发 - Keil 项目的导入与转换

◆ PTIDE 支持将 Keil MDK v5 项目导入并转换为 PTIDE 项目，依照以下步骤进行操作:

1. 开启 PTIDE 窗口，确认 PT_IDE PROJECTS 中无其他项目
2. 若有，请先在该项目上方点击右键，选择关闭项目
3. 点击 PT_IDE PROJECTS 或 OPERATIONS 栏位中的 Import Project
4. 在弹出的对话框中，选择 MDK arm projects
5. 选择欲导入的 Keil 项目文件 (.uvprojx)
6. VSCode 右下方会跳出项目储存位置选择提示
   a. 选择"是"，则导入的项目会储存在与 Keil 项目相同的目录下
   b. 选择"否"，则需指定新的项目储存路径，且需要自行建立项目文件夹
7. PTIDE 项目建立完成后，VSCode 右下方会跳出工作区转换提示，请点击确定
8. 此时 PT_IDE PROJECTS 区域中已建立新的项目，内容与导入的 Keil 项目相同，且已可以进行 Build & Program Flash 操作
   a. Project Resources 会依 Keil 项目中的文件结构建立虚拟目录，虚拟目录中的所有文件皆与原 Keil 项目相同，在 PTIDE 修改项目中的任意文件，均会同步更新至原 Keil 项目
   b. Builder Configuration 会自动设置为 Keil AC5
   c. Flasher Configuration 中的 Erase Type 预设为 full_erase

◆ 项目客制化设置 (承接上述流程)

1. 将原 Keil 项目中须修改的用户文件 (如 main.c 和 main.h) 复制到该 PTIDE 项目的储存位置中

2. 点击 VSCode 左侧[资源管理器]，即可以在工作区中看到刚复制的文件
3. 回到 PTIDE 窗口
   a. 于 Project Resources 中移除原 Keil 项目中须修改的用户文件，并将刚复制的文件加入对应的虚拟文件夹中
   b. 于 Project Attributes 中，根据需要更新 Include Directories 以确保项目能正确编译
4. 完成上述步骤后，可以点击 PT_IDE PROJECTS 区域中项目名称上方的 Build (F7) / Rebuild (Ctrl + Alt + F7) 按钮进行构建，再点开所有须修改的用户文件，即可以在 PTIDE 中确认该文件的引用路径是否正确无误
5. 确认无误后，即可以开始对用户文件进行修改与开发
6. 若需要创立新文件，请参考 Section 4.1, “建立新项目 (new project)” 小节中 "创立并引用新文件" 的说明进行操作

◆ 工具链转换 AC5 → GCC (接续上述流程)

1. 安装 Keil Pack
   a. 请先至普诚科技官网(https://www.princeton.com.tw/)下载所需的 Keil Pack 文件
   b. 于 PTIDE 窗口中，点击 PT_IDE PROJECTS 栏位中 Chip Support Package 旁的 Install CMSIS Chip Packet 按钮，选择下载好的 pack 文件并安装
   c. 点开 Chip Support Package 下拉选项，在安装后的文件旁点击 Select Device 按钮，选择目标设备，完成设备设置
2. 切换工具链
   a. 点击 Builder Configuration 旁的 Select Toolchain 按钮
   b. 在弹出的对话框中，选择 GNU Arm Embedded Toolchain GCC
3. 新增链接器脚本 (Linker Script)
   a. 于 Builder Configuration 底下的 Linker Script File Path 中，新增适当的链接器脚本文件 (xxx.ld)
   b. 文件位置: .pack\Princeton\ "Keil Pack 名称" \Device\ "目标 MCU 型号" \GCC\Source
4. 取代特定文件，以确保项目能正确编译
   a. 移除 Project Resources 中的 retarget.c 与 startup_xxx.s 文件
   b. 对应新增 retarget.c 与 startup_xxx.c 文件
   c. 文件位置: .pack\Princeton\ "Keil Pack 名称" \Device\ "目标 MCU 型号" \GCC\Source
5. 此时即完成工具链转换，将鼠标移到 PT_IDE PROJECTS 区域中项目名称上方，点击 Build

(F7) / Rebuild (Ctrl + Alt + F7) 按钮进行构建

6. 当构建成功后，即表示完成了完整的 Keil 项目导入与转换流程，后续即可在 PTIDE 中进行开发与调试

5 程序烧录

PTIDE 支持使用 OpenOCD 进行程序烧录，以下说明如何使用 PTIDE 进行烧录操作：

1. 确认 PTEM 调试器已连接至计算机，并且调试器已正确连接至目标设备
2. 点击 Rebuild (Ctrl + Alt + F7) 按钮，并确保该项目已成功构建，产生可执行文件 (HEX / BIN)
3. 下拉 Flasher Configuration
   a. 根据需要，点击 Erase option 旁的 Modify 按钮，选择适当的擦除类型 (full_erase / sector_erase / no_erase)
   b. 点击 Debug Probe Serial 旁的 Modify 按钮，由下拉菜单当中选择所需的 PTEM 设备
   c. 用户可以根据需求，调整烧录文件。预设为该项目所产生的 HEX 文件，若需更改，请点击 Program File 旁的 Modify 按钮，选择其他文件
4. 将鼠标移到 PT_IDE PROJECTS 区域中项目名称上方，点击 Program Flash (Ctrl + Alt + D) 按钮进行烧录操作
5. 烧录过程中，请注意 PTIDE 窗口下方的输出窗口，以确认烧录是否成功完成

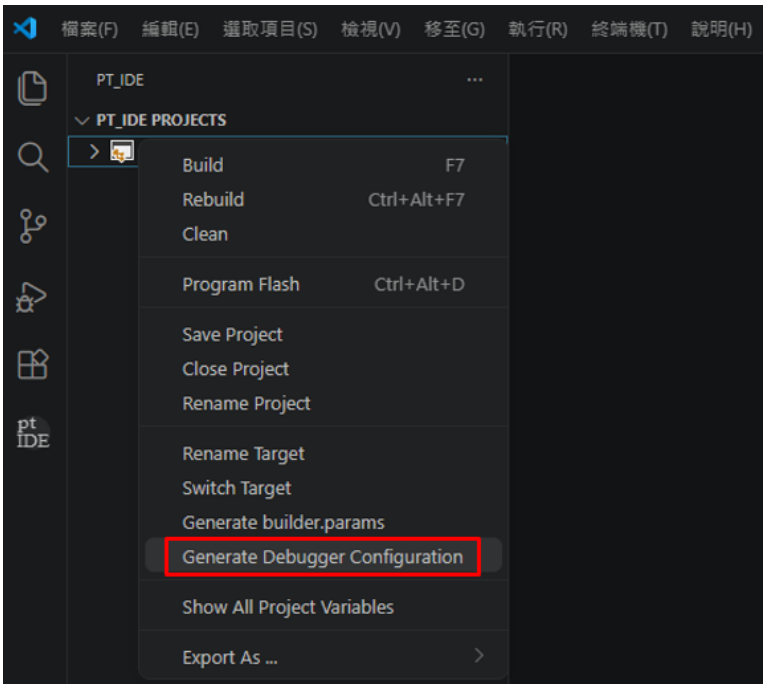
6 调试功能

6.1 如何进入调试模式

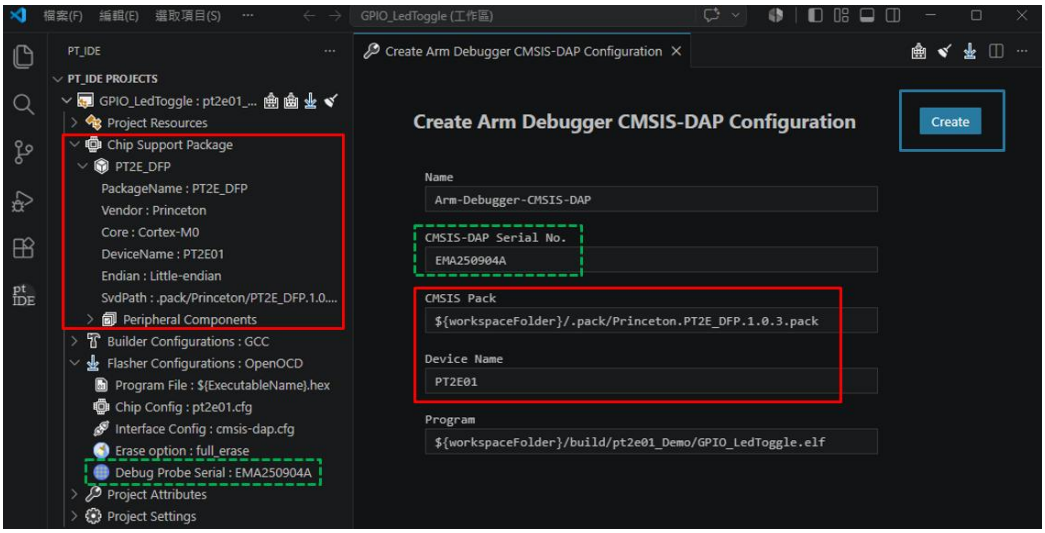
PTIDE 支持使用 Arm-Debugger 延伸模块进行调试操作，以下说明如何使用 PTIDE 进行调试：

1. 确认 PTEM 调试器已连接至计算机，并且调试器已正确连接至目标设备
2. 开启欲调试的 PTIDE 项目
3. 确认 Flasher Configuration 中的 Debug Probe Serial 已选择正确的 PTEM 设备
4. 生成 launch.json 文件
   a. 生成 launch.json 文件前，请先确保此项目已安装 Keil Pack，并选择正确的目标设备。安装方式请参考 Section 4.1, “建立新项目 (new project)” 小节中 "安装 Keil Pack" 的说明进行操作
   b. 将鼠标移到 PT_IDE PROJECTS 区域中项目名称上方，右键点击，选择 Generate

Debugger Configuration

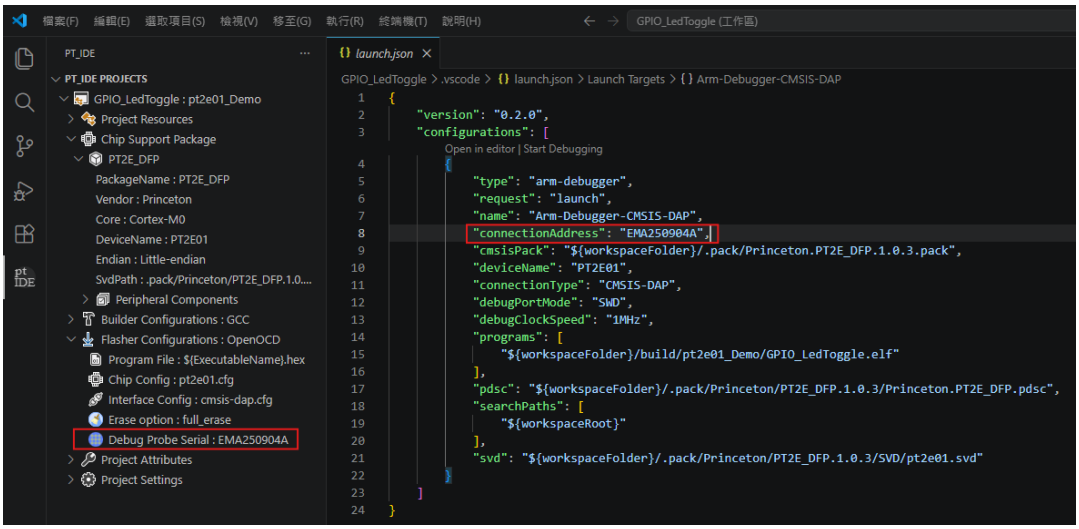


c. 显示 Create Arm Debugger CMSIS-DAP Debug Configuration 窗口，根据需求修改调试器序号后，点击 Create



d. 最终生成 launch.json 文件

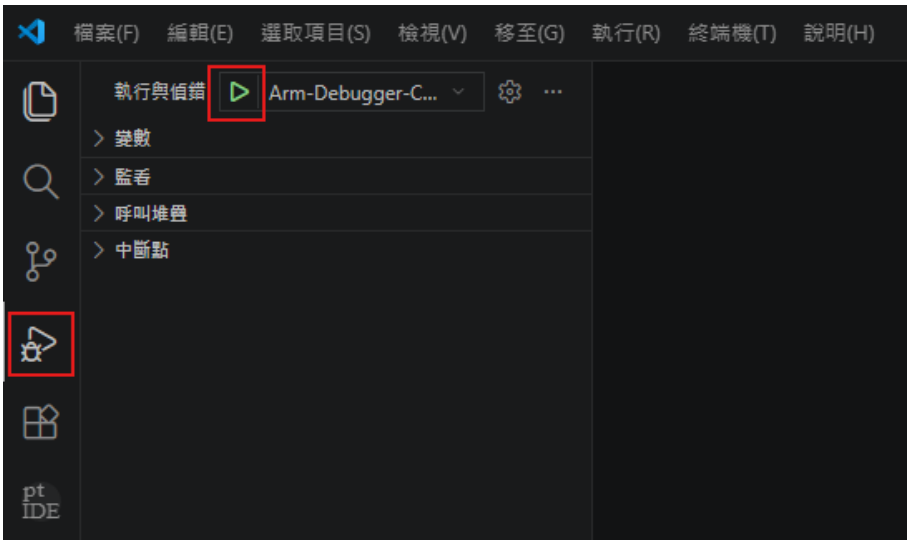
5. 若项目已生成 launch.json 文件，须注意调试器序号是否正确



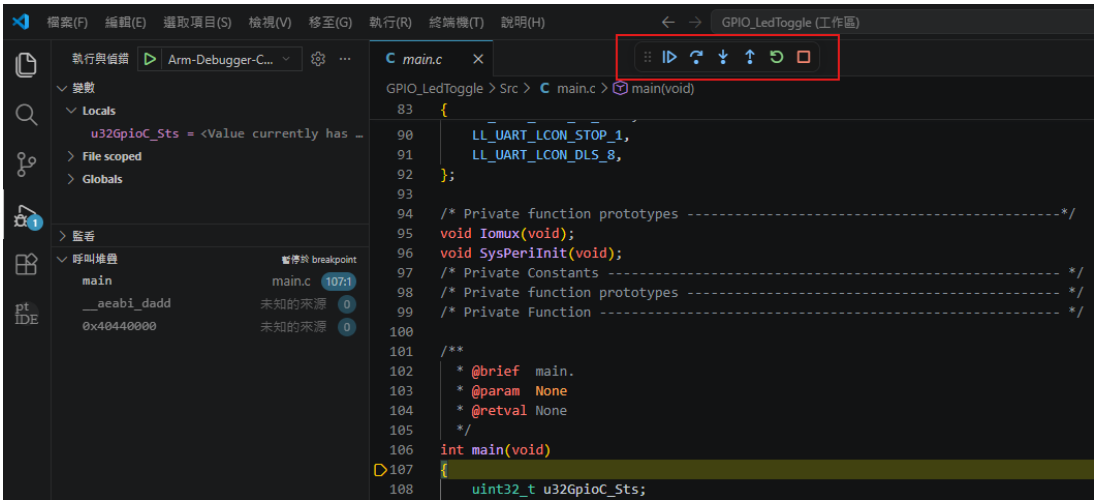
6. 开始调试前，请先点击 Rebuild (Ctrl + Alt + F7) 按钮，并确保该项目已成功构建，再点击 Program Flash (Ctrl + Alt + D) 按钮，将可执行文件烧录至目标设备

7. 点击 VSCode 左侧调试图标，进入调试窗口

8. 在[调试窗口]中，点击绿色按钮 (Start Debugging) 或按下 F5 键，开始调试程序。若为第一次调试，可以参考 Section 6.2, “第一次调试注意事项” 小节相关内容



9. 看到[调试工具栏](如红框中所示)，表示成功进入调试模式，工具栏由左至右分别为:



a. 继续 / 暂停 (Continue / Pause): 继续执行程序直到下一个断点 / 暂停执行程序

b. 不进函数 (Step Over): 执行当前行并停在下一行

c. 逐步执行 (Step Into): 进入函数内部并停在第一行

d. 跳离函数 (Step Out): 执行完当前函数并停在该函数的下一行

e. 重新启动 (Restart): 停止并重新启动调试程序

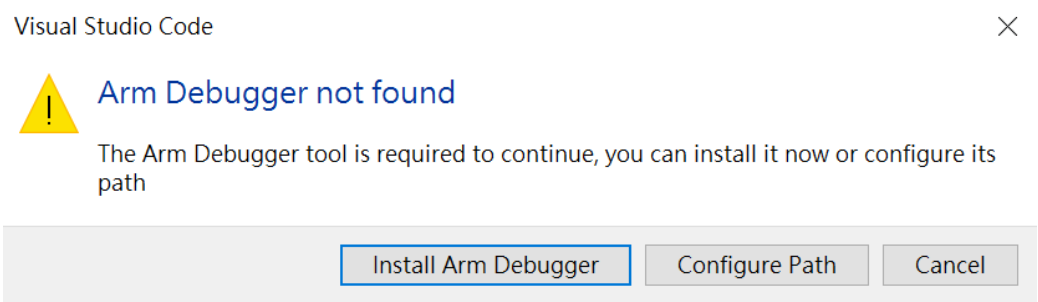
f. 停止 (Stop): 停止调试并结束调试模式

PTEM 设备	序号
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 之后	EML250804A or EML260601A
PTEM-V1AIR FW_V_1.0.3 之后	EMA250904A or EMK251114A
PTEM-V1	EMX260401A

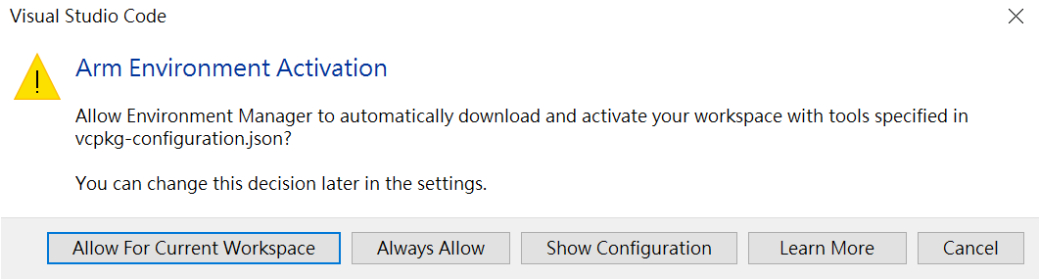
6.2 第一次调试注意事项

本章节纪录第一次在 PTIDE 中进入调试模式，VSCode 可能会跳出的提示:

1. 提示安装 Arm Debugger，请点击安装

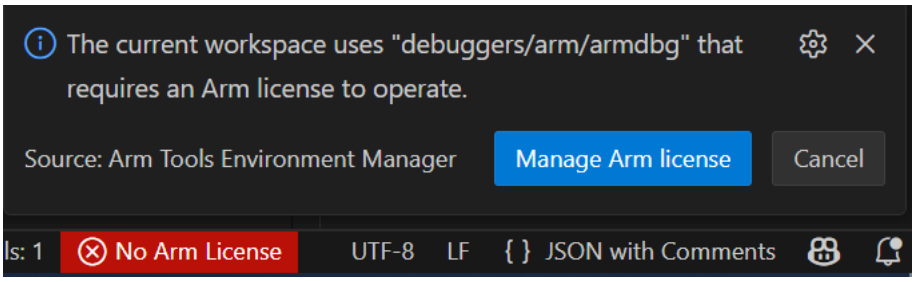


2. 提示安装必要的调试组件，请点击允许

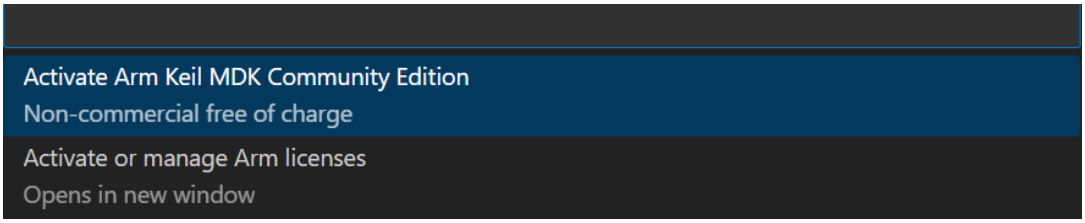


3. 提示需安装 Arm License

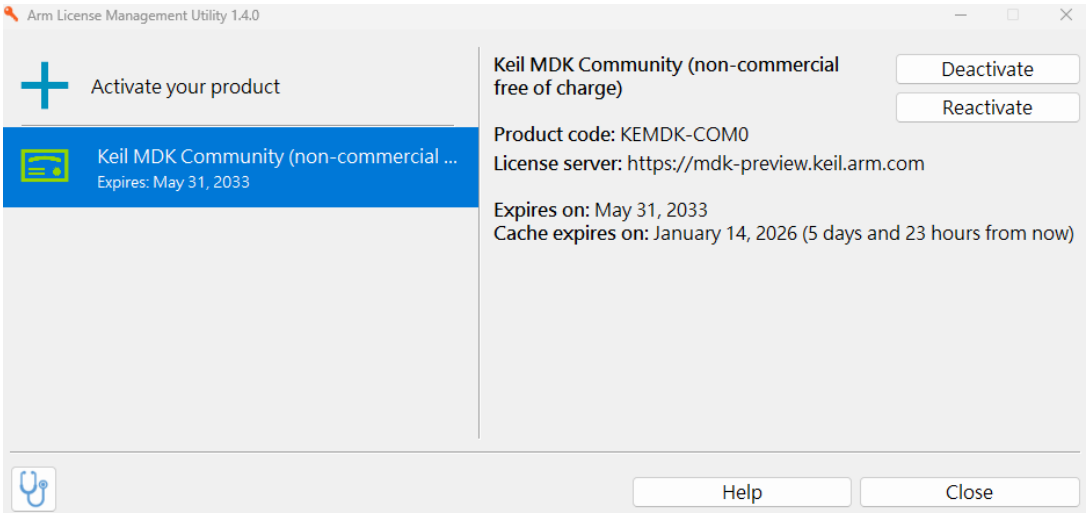
a. 点击 Manage Arm License 按钮或红底 No Arm License 警示



b. 下拉菜单中点击 Activate Arm Keil MDK Community Edition



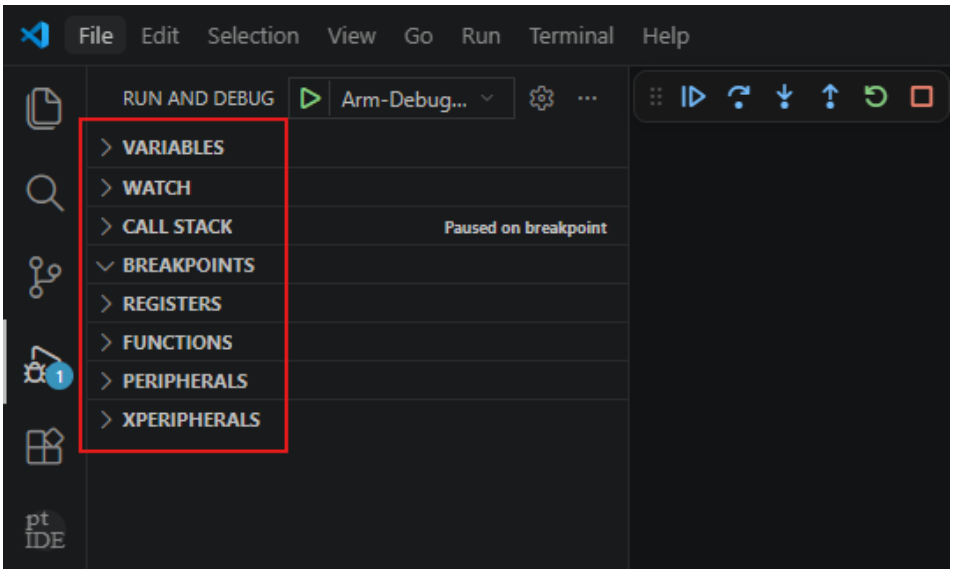
c. 在弹出窗口中点击 Keil MDK Community，再点击 Reactivate 按钮



6.3 监看功能

进入 debug mode 后，用户可以通过 VSCode 左侧监视窗口，了解调试过程中程序状态变化。以下介绍各个窗口的功能:

- VARIABLES (变量): 显示目前作用域内的所有变量及其值，用户可以展开对象或数组以查看其内部结构
- WATCH (监看): 用户可以手动新增变量或表达式，以便在调试过程中持续监控其值的变化
- CALL STACK (调用堆栈): 显示目前线程的调用堆栈，用户可以点击堆栈中的任意一层，以查看该层的代码位置
- BREAKPOINTS (断点): 列出所有已设置的断点，用户可以删除断点，或输入函数名称以此设置断点
- REGISTERS: 显示目标设备的 CPU 寄存器值，用户可以查看目前寄存器的内容，以便进行底层调试
- FUNCTIONS: 显示目前作用域内的所有函数，用户可以点击前往查看该代码位置和设置该函数为断点
- PERIPHERALS / XPERIPHERALS: 显示目标设备的外设寄存器，用户可以查看并修改寄存器数值，以便进行底层调试



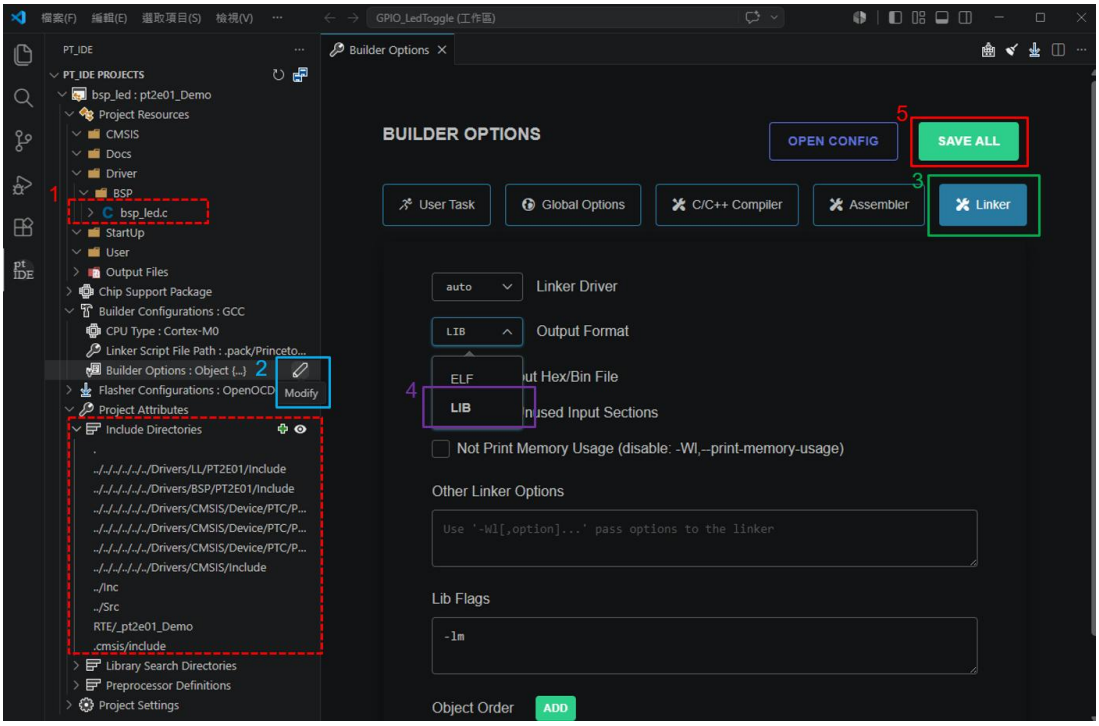
6.4 检视汇编语言

在每个 C 来源档中点击右键，选择 "Show Disassembly" 选项，即可检视对应的汇编程序代码。此功能在一般模式下即可使用，无需进入调试模式。

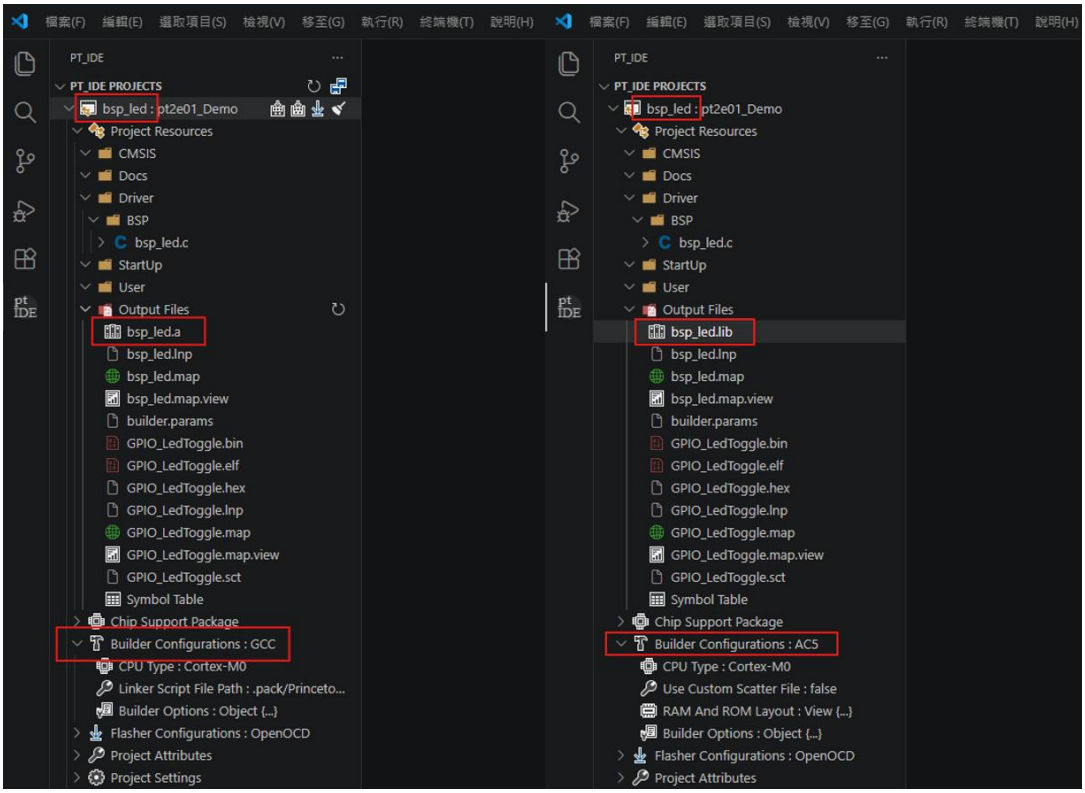
7 其他功能

7.1 静态库生成

1. 建构一个静态库项目
   - 仅包含目标(.c)文件
   - 该目标(.c)文件所需的所有头文件(.h)皆需要加载至 Project Attributes 的 Include Directories 中
2. 下拉 Builder Configuration，点击 Builder Options 旁的 Modify 按钮
3. 点击 Builder Options 窗口中的 Linker 选项
4. 下拉 Output Format，选择 LIB
5. 点击 SAVE ALL



6. 此项目即完成建立，将鼠标移到 PT_IDE PROJECTS 区域中项目名称上方，点击 Build (F7) / Rebuild (Ctrl + Alt + F7) 按钮进行构建
7. 成功后即可以在 Output Files 看到对应的文件(GCC: .a / AC5: .lib / AC6: .lib)，该文件即为静态库文件，默认名称为此项目名称，用户可依需求修改名称



8. 此静态库文件即可以取代其他项目中的目标(.c)文件

8 文件版本纪录

表 1. 文件修订版本记录

日期	版本	修订内容
2026-07-15	V1.0	初版