匠芯创科技
ARTINCHIP

D12X 快速入门

国产自主　32 位 RISC-V

Version 1.1

June.12,2024

修订记录

版本	日期	修订人	修订说明
V1.0	2023-08-30	Guojun.dong	初版
V1.1	2024-06-12	Wang.tang	加入 VSCode 环境

版权声明

本文档是匠芯创科技（“ARTINCHIP”）的原创作品，匠芯创科技拥有该文档的全部版权。全部或部分复制必须获得匠芯创科技的书面批准，并向版权所有人明确确认。凡侵犯本公司版权等知识产权的，本公司将保留依法追究其法律责任的权利。

在法律允许的范围内，在此声明：使用前请仔细阅读合同条款和条件以及相关说明，并严格遵守本文档中的说明。匠芯创科技不对不当行为的后果（包括但不限于过电压、超频或温度过高）承担任何责任。

匠芯创科技提供的信息仅作为参考或典型应用，本文档中的所有声明、信息和建议不构成任何明示或暗示的担保。匠芯创科技保留随时更改电路设计和/或规格的权利，恕不另行通知。

客户应全权负责获得实施解决方案/产品可能需要的第三方许可，匠芯创科技不承担任何与第三方许可相关的许可费或特许权使用费。对于任何要求的第三方许可证所涵盖的事项，匠芯创科技不承担任何保证、赔偿或其他义务。

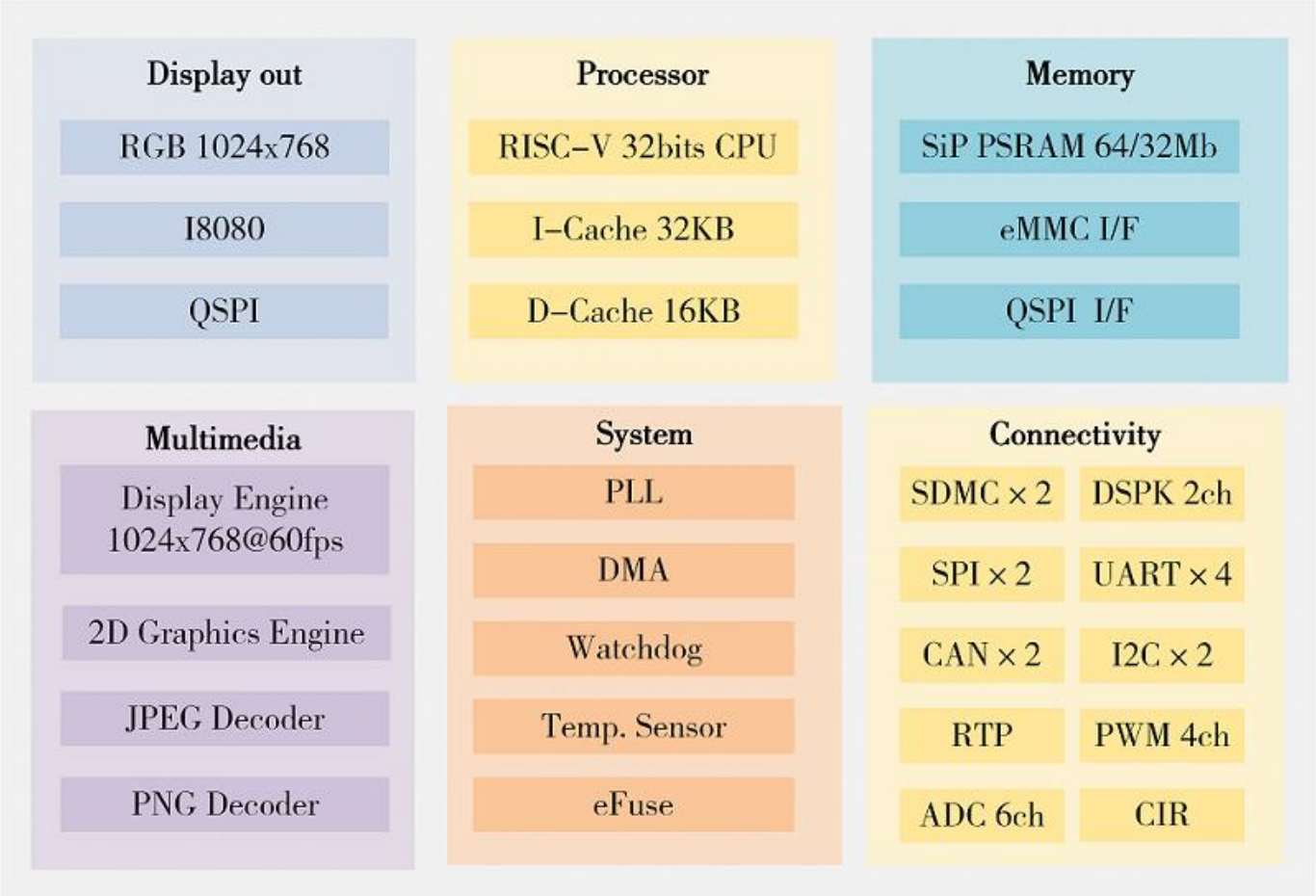
凡以任何方式直接或间接使用本文档资料者，视为自愿接受本文档声明的约束。

目录

1. SOC

D12x 是一款基于 RISC-V 的高性能、国产自主、工业级高清显示与智能控制 MCU，配备强大的 2D 图形加速处理器、PNG/ JPEG 解码引擎、丰富的接口，支持工业宽温，具有高可靠性、高开放性，可广泛应用于工业自动化控制、串口屏等智慧工业和智慧家居领域。

1. 1. 功能框图



2. 开发板

2.1. D122BB-Demo-V1.0

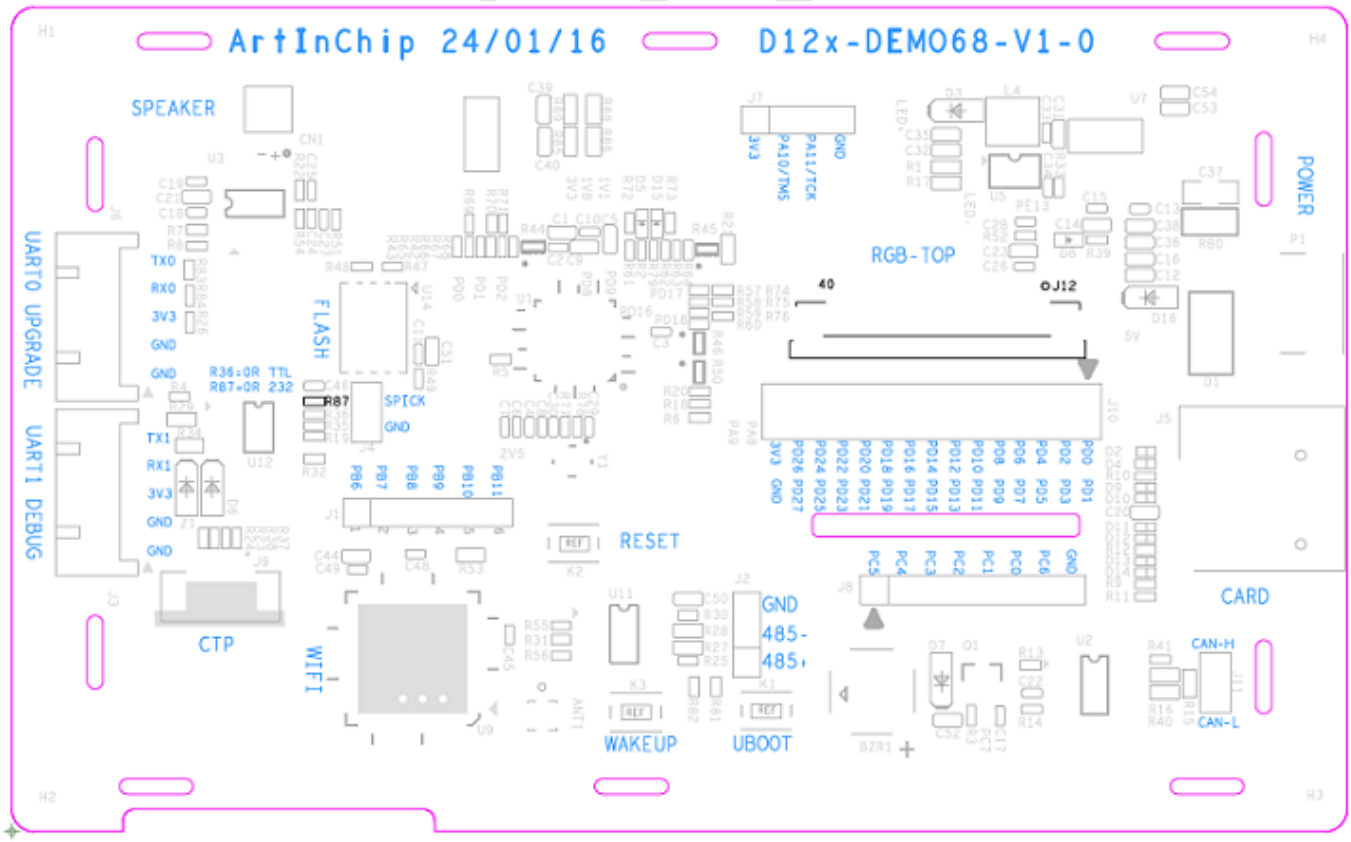
2.1.1.开发版标识

D12x-Demo68-V1.0

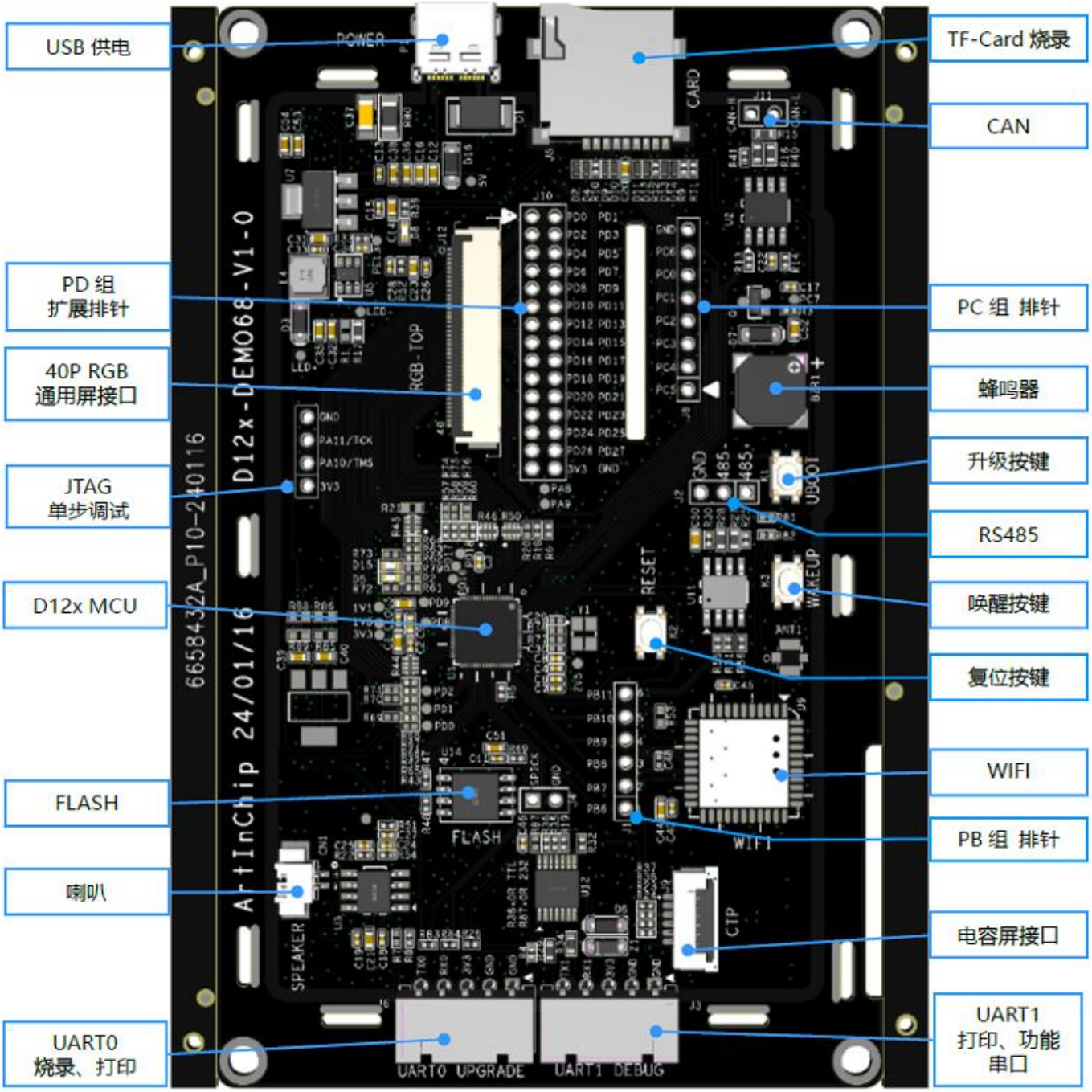
2.1.2.规格

- RGB888 x 480 x 272
- 8 MB PSRAM + 16 MB NOR
- RS485 + RS232 + UART
- SDIO WiFi + TF-Card
- DSPK + Buzzer

2.1.3.器件布局



2.1.4.实物图



2.1.5.方案配置

方案的配置对应的是 target/D12x/demo68_nor/ 工程

编译选项：D12x_demo68-nor_rt-thread_helloworld_defconfig

固件：D12x_demo68-nor_v1.0.0.img

2.2. D122BB-HMI-V1.1

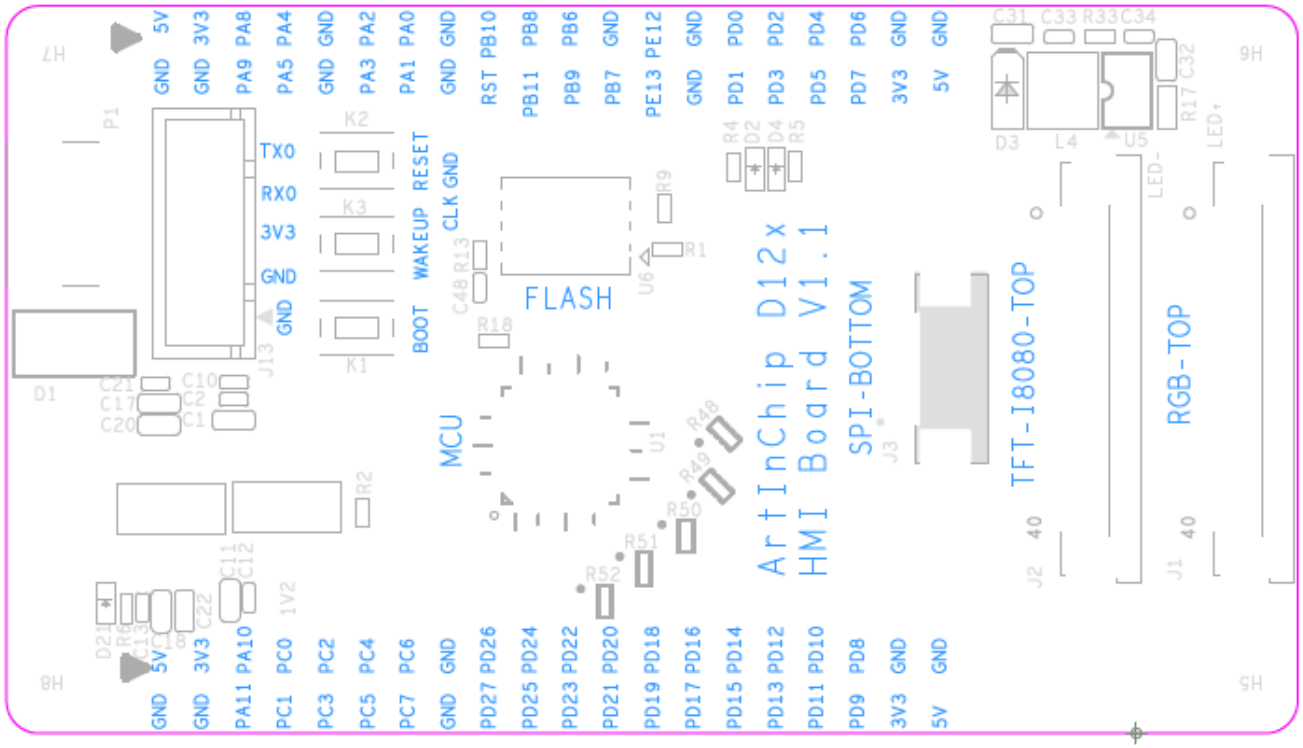
2.2.1.开发版标识

ArtInChip HMI Board V1.1

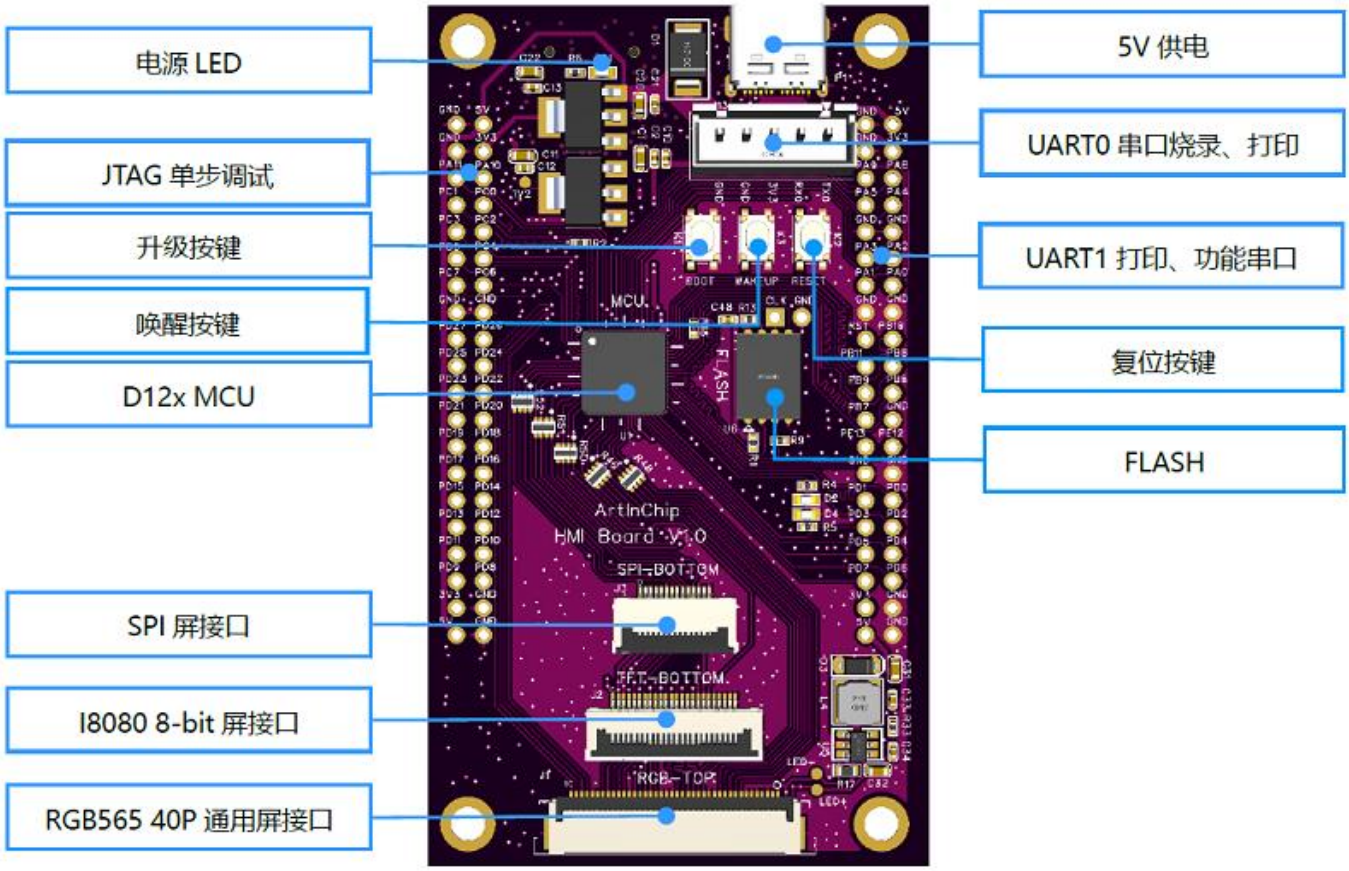
2.2.2.规格

- RGB888 x 480 x 272 + SPI + I8080
- 8 MB PSRAM + 16 MB NOR
- UART x 2
- TF-Card
- 2 x 扩展排针

2.2.3.器件布局



2.2.4.实物图



2.2.5.方案配置

方案的配置对应的是 target/D12x/hmi_nor/ 工程

编译选项：D12x_hmi-nor_rt-thread_helloworld_defconfig

固件：D12x_hmi-nor_v1.0.0.img

3. 快速编译指南

遵照以下快速开发指引编译和调试 SDK：

1. 下载代码仓库。
2. 准备 SDK 环境并编译 SDK。

   SDK 操作环境分为 Luban-Lite SDK 和 Baremetal，选择其中一种即可。
   每个 SDK 对应三种操作环境，分别为 VSCode，Windows 和 Linux 系统，可根据实际需求选择。

3. 烧写 SDK。
4. 使用 ArtInChip 刷机工具执行刷机。
5. 调试 SDK。

4. 下载代码仓库

ArtInChip 通过码云（gitee）提供 D12x 相关的仓库资源下载且全部开源，包括：

- Luban-Lite（RTOS）：git clone https://gitee.com/artinchip/luban-lite.git
- Baremetal（裸机）：git clone https://gitee.com/artinchip/baremetal.git
- 文档
  1. 官网在线阅览和下载资源：http://aicdoc.artinchip.com
  2. 从 Gitee 下载离线版本：git clone https://gitee.com/artinchip/d12x-doc.git
- 工具：git clone https://gitee.com/artinchip/tools.git

5. 编译 SDK

对于 D12x，ArtInChip 提供下列 SDK 供用户选择：

- Luban-Lite 是 ArtInChip 基于 RT-Thread 深度开发的嵌入式实时系统。
- Baremetal 是 ArtInChip 的嵌入式裸机系统。

本章节主要介绍如何在 Linux 、Windows 和 VSCode 上快速搭建环境和编译固件。关于 eclipse 等 IDE 工具的使用，可参考详细文档。

5.1. Luban-Lite SDK

5.1.1.环境准备

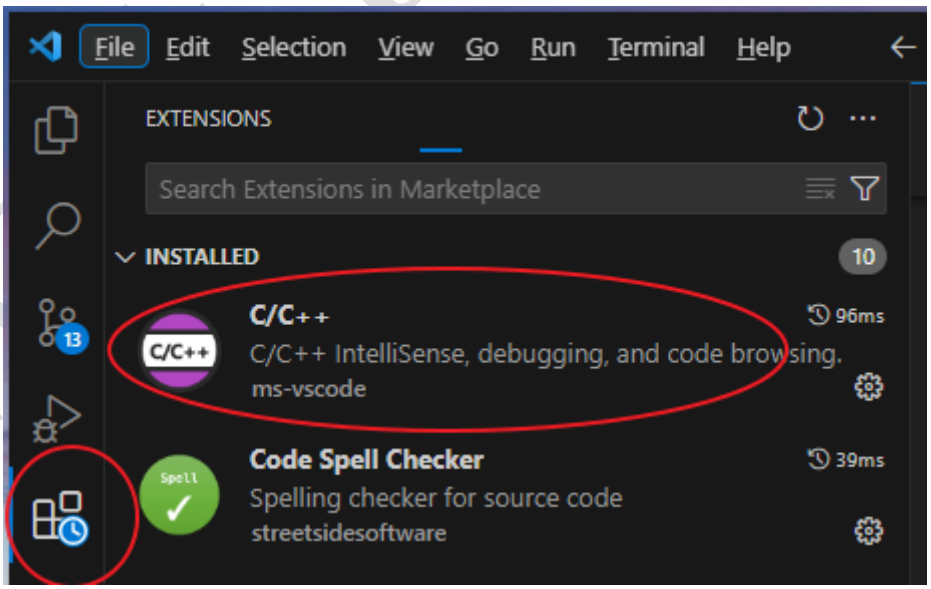
5.1.1.1. VSCode

VSCode 是一款开源、免费、跨平台的源代码编辑器，由 MicroSoft 开发，特定是轻量级、高性能、可扩展。

VSCode 环境可以实现 Luban-Lite 和 Baremetal 全流程的开发，包括代码编辑、编译、调试和烧写。在 VSCode 中完成 SDK 编译的详细流程如下：

5.1.1.1.1. 安装 C/C++ IntelliSense 插件

VSCode 中需要进行 C/C++ 语言的 Debug，必须先安装插件 C/C++ IntelliSense，如下图所示。

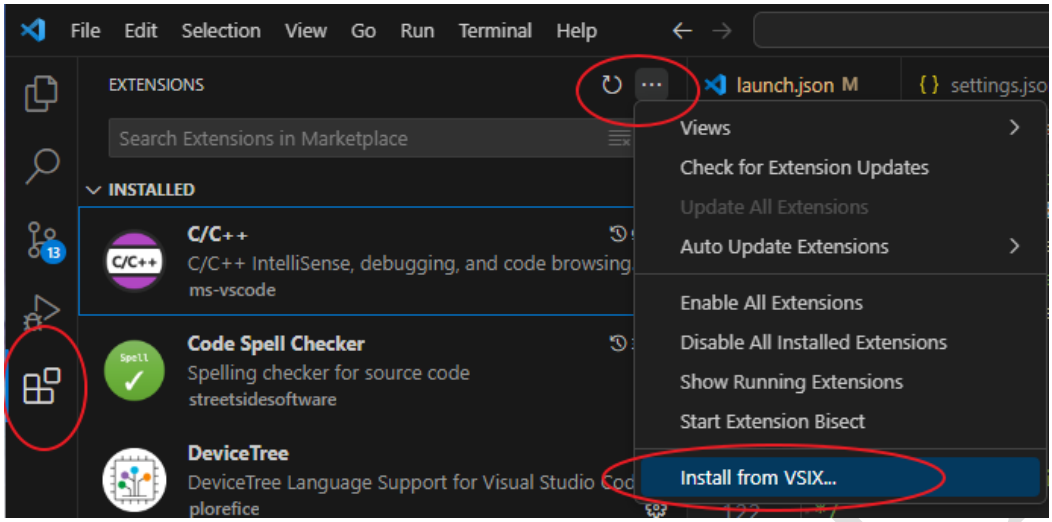


安装方法如下：

- 如果 PC 有联网，直接点击 Install 即可完成安装。安装完成后的效果如上图所示。
- 如果 PC 没有联网，需要手动安装，详细步骤如下：

1. 在 VSCode 官网下载 C/C++ IntelliSense 插件的安装文件，后缀名为 .vsix。

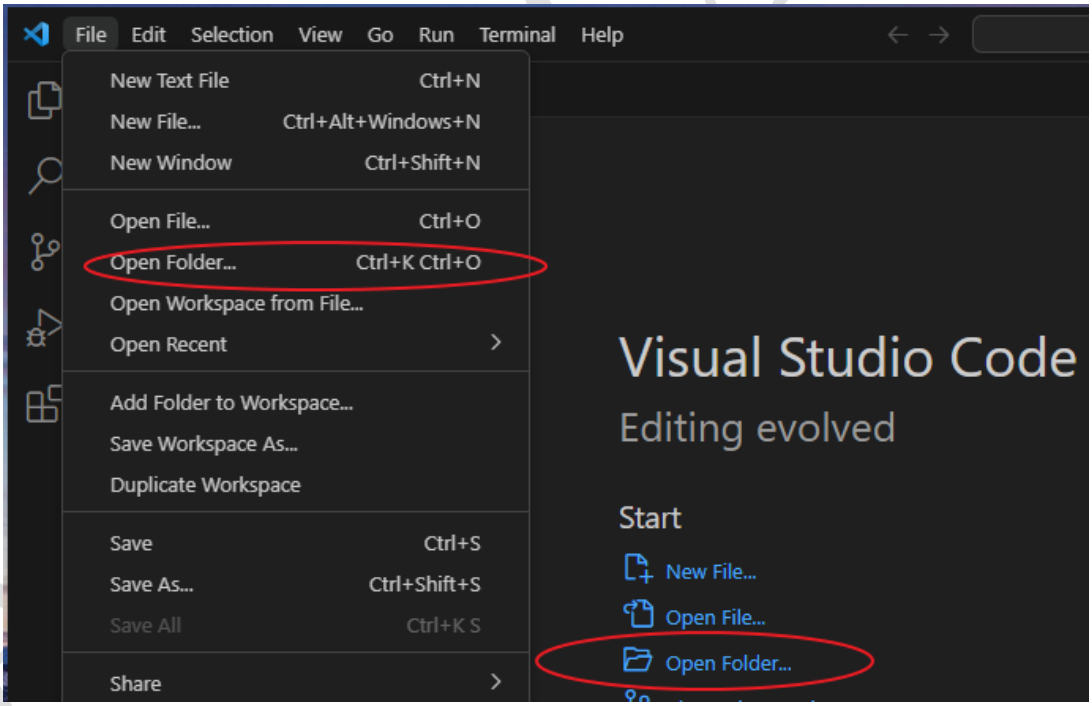
2.在 VSCode 的插件管理界面，选择从 VSIX 安装插件，如下图所示：



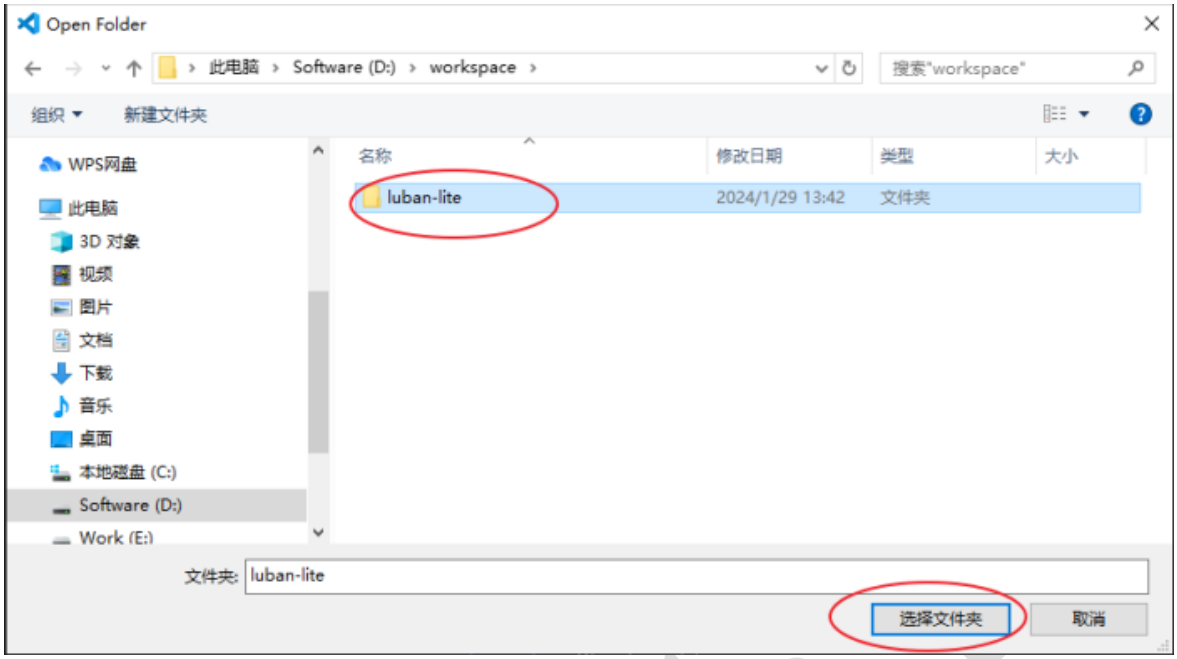
5.1.1.1.2. 打开 Luban-Lite 工程

与 Eclipse 工具的工程配置不同，VSCode 没有严格的"工程"概念。只要打开一个目录，就相当于创建了一个工程。

使用下图中的任意一种方式即可打开一个目录：



在弹出的文件夹浏览窗口中选择并点击 luban-lite 根目录文件夹：



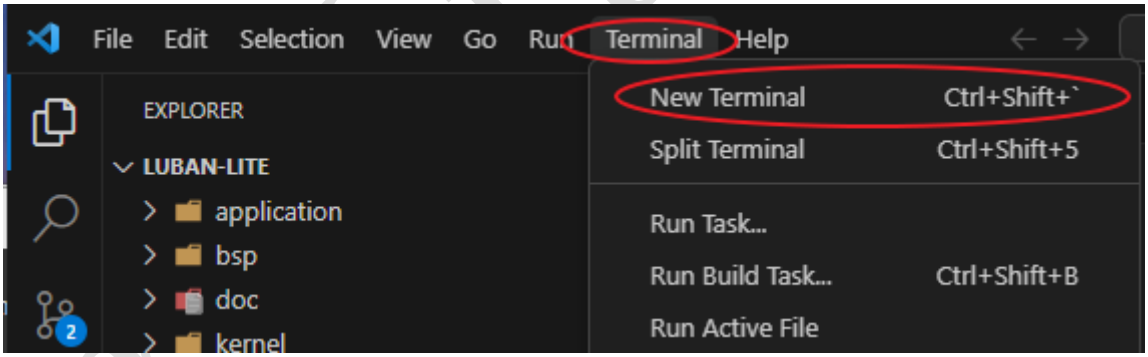
注意

VSCode 必须且只能在打开 luban-lite 根目录的前提下，才能完成以下的编译、调试、烧写操作。

5.1.1.1.3. 编译 Luban-Lite

执行下列步骤编译 Luban-Lite：

1. 打开 VSCode 终端（快捷键 Ctrl + Shift + `）：



2. 在 VSCode 终端窗口中输并执行 win_cmd.bat。

3. 用 list 命令查询，查询结果如下图所示：

```
E:\AIC\luban-lite\luban-lite>list
scons: Reading SConscript files ...
Built-in configs:
  0. d12x_demo68-nand_baremetal_bootloader
  1. d12x_demo68-nand_rt-thread_helloworld
  2. d12x_demo68-nor_baremetal_bootloader
  3. d12x_demo68-nor_rt-thread_helloworld
  4. d12x_hmi-nor_baremetal_bootloader
  5. d12x_hmi-nor_rt-thread_helloworld
  6. d13x_demo88-nand_baremetal_bootloader
  7. d13x_demo88-nand_rt-thread_helloworld
  8. d13x_demo88-nor_baremetal_bootloader
  9. d13x_demo88-nor_rt-thread_helloworld
 10. d13x_kunlunpi88-nor_baremetal_bootloader
 11. d13x_kunlunpi88-nor_rt-thread_helloworld
 12. d21x_demo128-nand_baremetal_bootloader
 13. d21x_demo128-nand_rt-thread_helloworld
 14. g73x_demo100-nor_baremetal_bootloader
 15. g73x_demo100-nor_rt-thread_helloworld
```

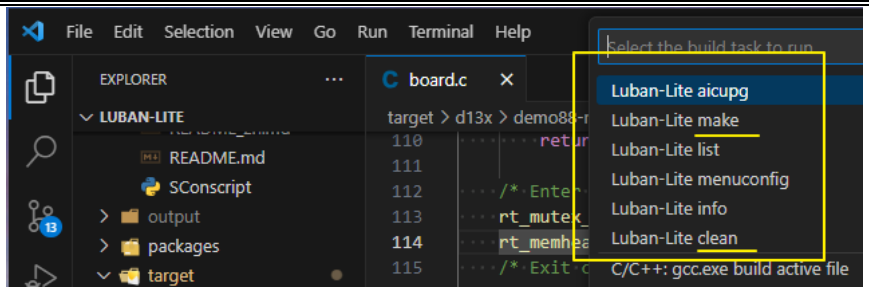
4. 选择开发板或者方案所对应的配置编号，执行 lunch 命令。

   例如：当前环境是使用的是 D12X_demo68-nor_rt-thread_helloworld 方案，其配置的编号是 3，执行命令 lunch 3。

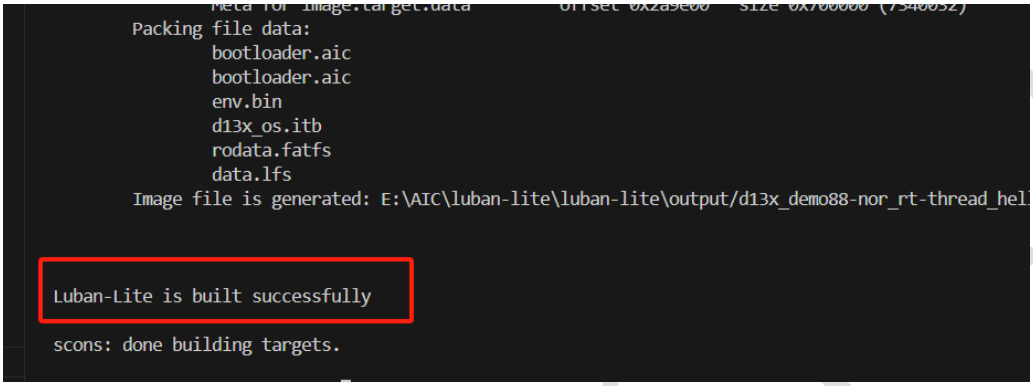
```
E:\git\luban-lite>list
scons: Reading SConscript files ...
Built-in configs:
  0. d12x_demo68-nand_baremetal_bootloader
  1. d12x_demo68-nand_rt-thread_helloworld
  2. d12x_demo68-nor_baremetal_bootloader
  3. d12x_demo68-nor_rt-thread_helloworld
  4. d12x_hmi-nor_baremetal_bootloader
  5. d12x_hmi-nor_rt-thread_helloworld
  6. d13x_demo88-nand_baremetal_bootloader
  7. d13x_demo88-nand_rt-thread_helloworld
  8. d13x_demo88-nor_baremetal_bootloader
  9. d13x_demo88-nor_rt-thread_helloworld
 10. d13x_kunlunpi88-nor_baremetal_bootloader
 11. d13x_kunlunpi88-nor_rt-thread_helloworld
 12. d21x_demo128-nand_baremetal_bootloader
 13. d21x_demo128-nand_rt-thread_helloworld
 14. g73x_demo100-nor_baremetal_bootloader
 15. g73x_demo100-nor_rt-thread_helloworld

E:\git\luban-lite>lunch 3
scons: Reading SConscript files ...
Load config from target\configs\d12x_demo68-nor_rt-thread_helloworld_defconfig
```

5. 使用以下任意方式触发编译：
   a) 在 VSCode 终端中输入 m 命令
   b) 通过 VSCode 的快捷命令 Ctrl + Shift + B 然后选择 Luban-Lite make。
6. 选择以下任意方式 clean 工程。
   a) 在 VSCode 终端中输入 c 命令
   b) 通过 VSCode 的快捷命令 Ctrl + Shift + B 然后选择 Luban-Lite clean

   使用快捷命令执行 make、clean 的方法如下图：



编译成功后结果如下：



其所生成的镜像文件如下：

luban-lite\output\D12X_demo88-nor_rt-thread_helloworld\images\D12X_demo88-nor_v1.0.0.img

5.1.1.2. Windows 系统

遵照以下流程编译 Luban-Lite：

1. 在 Luban-lite/tools/env/ 目录中选择并双击 Luban-lite/win_env.bat 或者 Luban-lite/win_cmd.bat 文件。Windows 下对应的各种工具已经存放在 Luban-lite/tools/env/ 目录中，不需要安装。
2. 解析配置名，例如：D12x_demo68-nor_rt-thread_helloworld

   - D12x:表示芯片类型。
   - Demo68：表示芯片引脚为 68pin 的 demo 板。
   - nor:表示外挂 flash 为 nor flash。
   - rt_thread:表示为 rt_thread 系统(相对应 baremetal 为裸机).
   - helloworld:表示为 app 配置（相对应 bootloader 为 boot 配置）。

```
~/luban-lite$ scons --list-def
scons: Reading SConscript files ...
Built-in configs:
  0. D12x_demo68-nand_baremetal_bootloader
  1. D12x_demo68-nand_rt-thread_helloworld
  2. D12x_demo68-nor_baremetal_bootloader
  3. D12x_demo68-nor_rt-thread_helloworld
  4. D12x_hmi-nor_baremetal_bootloader
  5. D12x_hmi-nor_rt-thread_helloworld
  6. d13x_demo88-nand_baremetal_bootloader
  7. d13x_demo88-nand_rt-thread_helloworld
  8. d13x_demo88-nor_baremetal_bootloader
  9. d13x_demo88-nor_rt-thread_helloworld
 10. d13x_kunlunpi88-nor_baremetal_bootloader
 11. d13x_kunlunpi88-nor_rt-thread_helloworld
 12. d21x_demo128-nand_baremetal_bootloader
 13. d21x_demo128-nand_rt-thread_helloworld
 14. g73x_demo100-nor_baremetal_bootloader
 15. g73x_demo100-nor_rt-thread_helloworld

~/luban-lite$ scons --apply-def=9          //选择 3(D12X_demo68-nor_rt-thread_helloworld) 号配置
~/luban-lite$ scons                        //编译

Image file is generated: E:\git\luban-lite\output/D12X_demo68-nor_rt-
thread_helloworld/images/D12X_demo68-nor_v1.0.0.img

编译后的固件名称为 D12X_demo68-nor_v1.0.0.img
```

5.1.1.3. Linux 系统

Luban-Lite SDK 的开发可以在 Linux 系统中进行，Luban-Lite SDK 目前自动支持的 Linux 发行版为：

- Ubuntu 14.04、16.04、18.04、20.04、22.04
- CentOS 7.x、8.x

Ubuntu 的安装教程在网上有很多可参考，以及相关常见问题也可以通过搜索查找解决方案。

- Ubuntu 官方网站：http://www.ubuntu.com
- 中文地址为：http://www.ubuntu.org.cn/index_kylin
- 桌面版下载地址：http://www.ubuntu.com/download/desktop

5.1.1.3.1. 安装依赖

Luban-Lite 的开发环境中，还需要安装一些依赖包：

- Python2： 用于编译
- scons：自动化构建工具
- Python3 + pycryptodomex： 用于打包和签名

在命令行中安装以上依赖的方法：

```
~/luban-lite$ cd luban-lite/
~/luban-lite$ sudo apt install scons
~/luban-lite$ sudo apt install pip
~/luban-lite$ cd tools/env/local_pkgs/
~/luban-lite$ tar xvf pycryptodomex-3.11.0.tar.gz
~/luban-lite$ cd pycryptodomex-3.11.0
~/luban-lite$ sudo python3 setup.py install
```

5.1.1.3.2. 编译 Luban-Lite

请参考 3.2.2.1 编译 luban-lite 章节。

5.2. Baremetal SDK

Baremetal 的环境准备和 Luban-Lite 的类似，可参考 Luban-Lite 的环境准备章节。

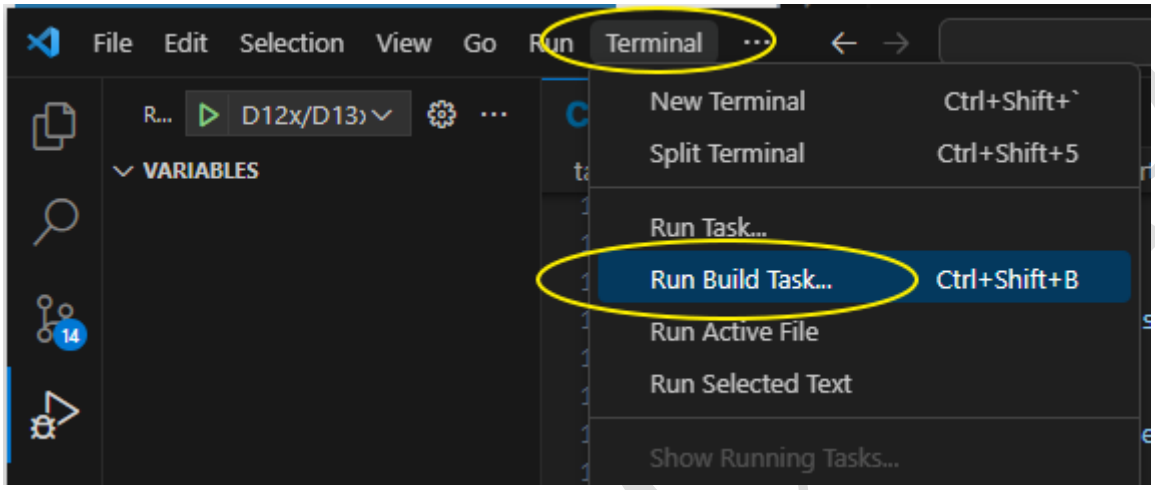
6. 烧写 SDK

Luban-Lite 的 OneStep 命令、 VSCode 的快捷命令中都已经集成了烧写功能。启动方法：

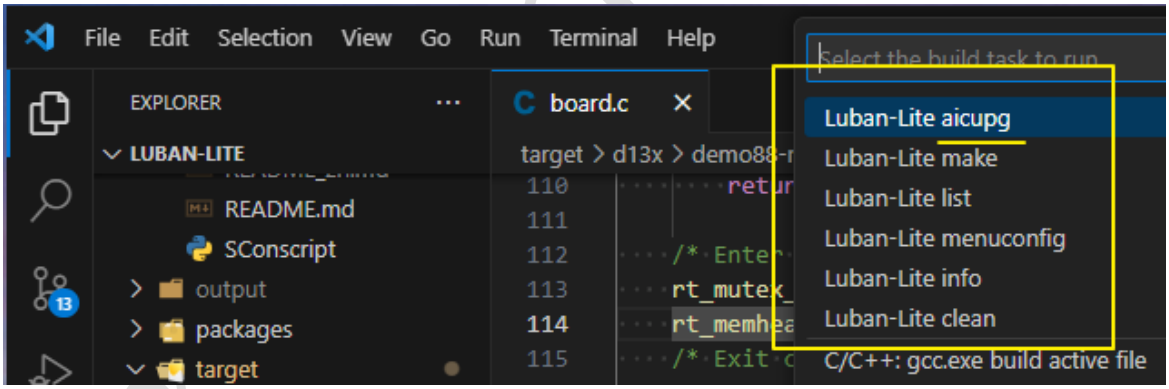
首先，让板子进入烧写模式

方式 1：OneStep 命令方式：在 VSCode 的终端中执行命令：aicupg

方式 2：VSCode 从界面中执行快捷命令的方式（Ctrl + Shift + B）



在弹出的命令列表中，选择 Luban-Lite aicupg :



如上图所示，Luban-Lite 还提供了其他快捷命令，包括：

1.list - 列出当前所有方案配置

2.menuconfig - 打开 menuconfig 配置界面

3.info - 查看当前的方案配置

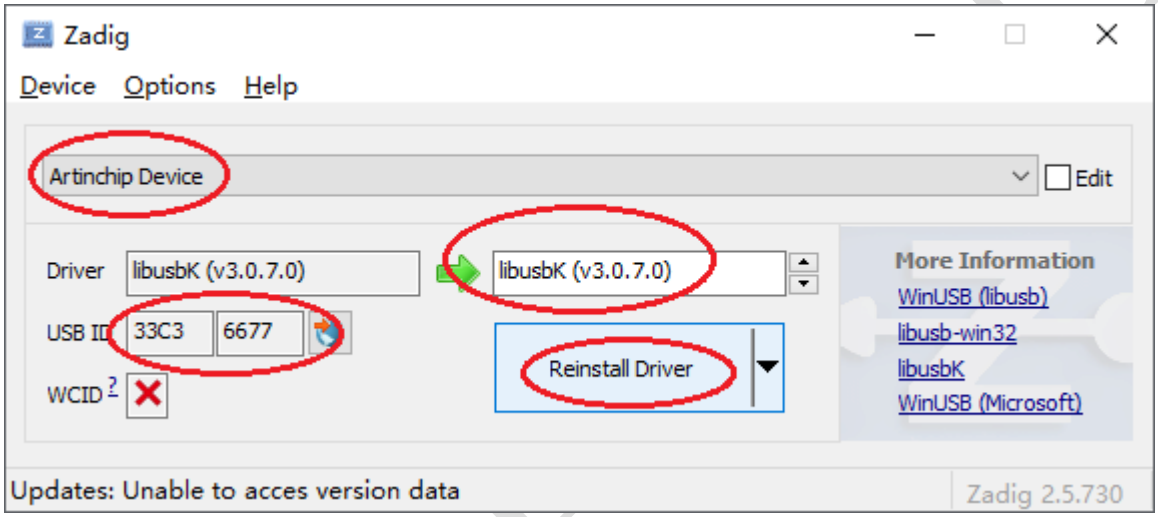
7. 刷机工具

ArtInChip 提供两组工具：

- AiBurn：单机调试刷机工具
- AiBurnPro：一拖八量产刷机工具

7.1. 驱动整理

AiBurn 通过 USB 烧录固件时需要 libusb 的支持，如果所用计算的 USB 驱动安装过于复杂而导致 ArtInChip 设备驱动安装异常时建议使用 tools 下的 Zadig 进行 USB 驱动的整理，方法如下图：



7.2. 刷机

AiBrun 的使用非常简单，选择编译好的镜像，在开发板进入烧写模式后点击“开始”按钮即可自动进行烧写，进入烧写模式有如下几种方式：

- 终端设备为空片，则上电直接进入 USB 烧写模式
- 启动时（上电或者按“重启键”）按住“烧录键”直接进入烧录模式
- 启动时短路存储介质造成读失败可以进入烧录模式（如果 SPINAND 的 4 和 5 脚）
- 终端设备非空片，如果能进入 终端，则执行命令 aicupg ，系统直接重启进入烧写模式

7.3. 串口调试

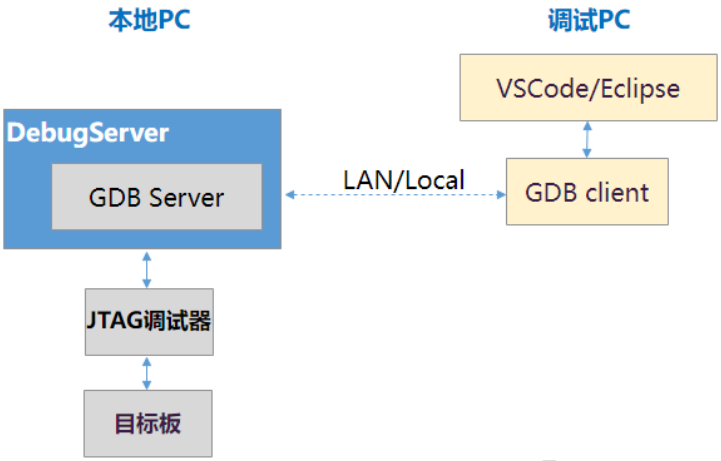
烧写镜像完成后可以通过串口进行信息的查看，默认的调试串口配置信息为：

- BaudRate：115200
- Data bits：8
- Stop bits：1
- Parity：None
- Flow control：None

8. 调试 SDK

8.1. 环境准备

JTAG 调试的整个物理环境可以抽象为：



上图涵盖两种网络环境：“本地 PC”和“调试 PC”可以是同一台 PC，也可以是局域网内不同的两台 PC。

使用 JTAG 调试需要准备的硬件：

1.板子上有 JTAG 插座，或者飞线引出了 JTAG 信号线，可以连接到 JTAG 调试器

2.JTAG 调试器，Luban-Lite 支持 CKLink 调试器 和 AIC JTAG 两种

3.并且保证板子和 JTAG 调试器的信号线正确连接，请参考调试器上的信号标识

使用 JTAG 调试需要准备的软件：

T-HeadDebugServer，调试器在 PC 端的代理，提供 GDB Server 调试服务。

AiBurn，ArtInChip 烧录软件，需要用到其中的 USB 烧写驱动。

以上两个软件请提前安装，安装过程涉及驱动安装所以都需要管理员权限；安装包可以在工具包中找到。

8.2. 两种场景

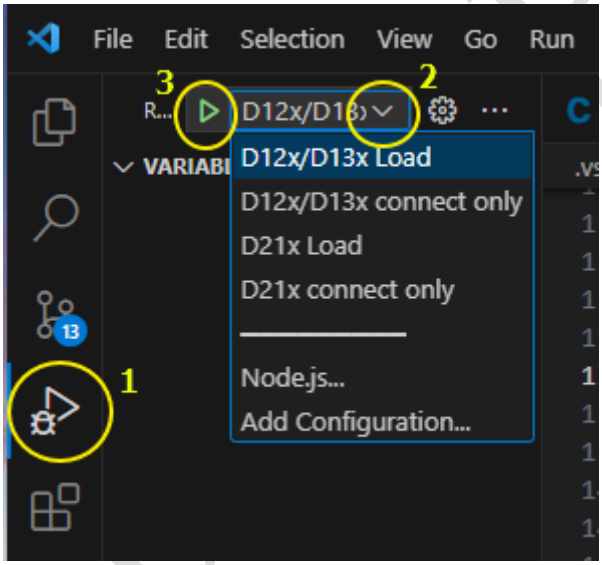
首先，要明确当前需要 JTAG 调试的场景是哪一种：

1. 板子刚执行完 PSRAM/DDR 的初始化，等待 JTAG 连接，Debug 配置选择执行： Dxx load
2. 板子上已经在运行一份镜像，中途用 JTAG 连接，Debug 配置选择执行： Dxx connect only

Luban-Lite 的 VSCode 配置中已经默认提供了四种 JTAG 选择，选择的方法：

SoC 型号	板子上无镜像	板子上已经在运行镜像
D21x	D21x load	D21x connect only
D12X/D12x	D12X/D12x load	D12X/D12x connect only

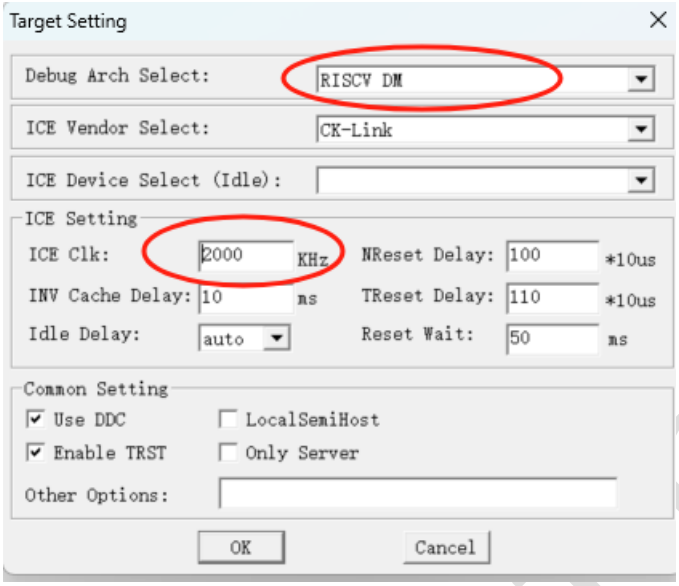
VSCode 中选择 Debug 配置的方法，从界面操作如下，选择合适的 Debug 配置（只需选择一次，VSCode 会记住上次的配置），然后点击箭头小图标（快捷键 F5）：



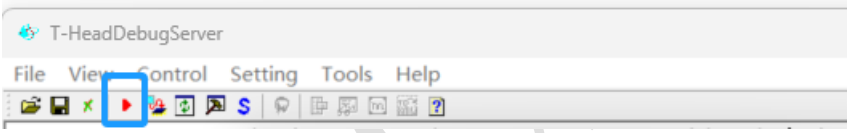
8.3. 软件配置

8.3.1. T-HeadDebugServer 配置

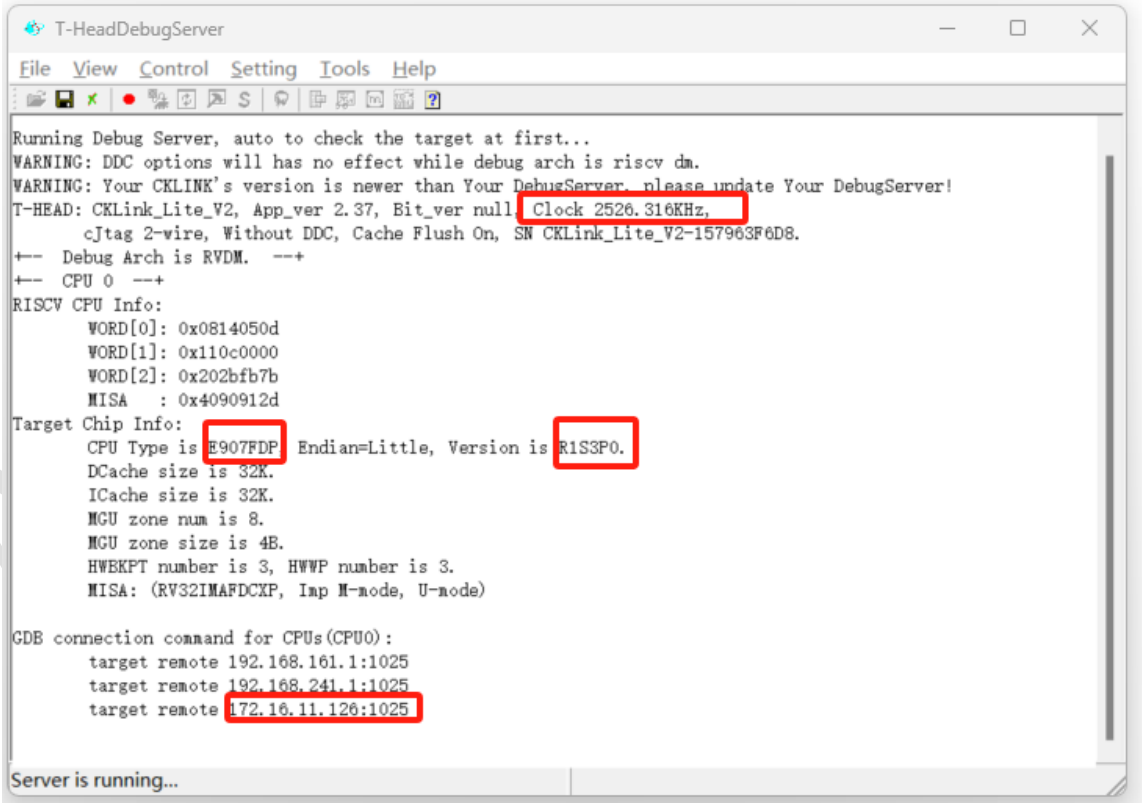
安装工具包中的 T-HeadDebugServer，其完成安装后，桌面会自动创建了一个 T-HeadDebugServer 的图标，运行 T-HeadDebugServer，首先要配置参数，在 Setting 目录下选择 target Setting（Debug Arch Select 必须选择 RISC-V DM，D1 需要选择 CSKY HAD）：



调试器正常连接，目标板上电，点击红色小三角按钮（或者 Control->RunDebugServer），开始连接设备：



正常情况下，会看到扫描出 CPU 信息如下：



注意：

1.通过 JTAG load elf 前，必须要先完成 PSRAM、或者 DDR 初始化，不然 JTAG 在写 PSRAM/DDR 时就会异常，使用 JTAG 口需要关闭 IIC、以及 Touch panel.

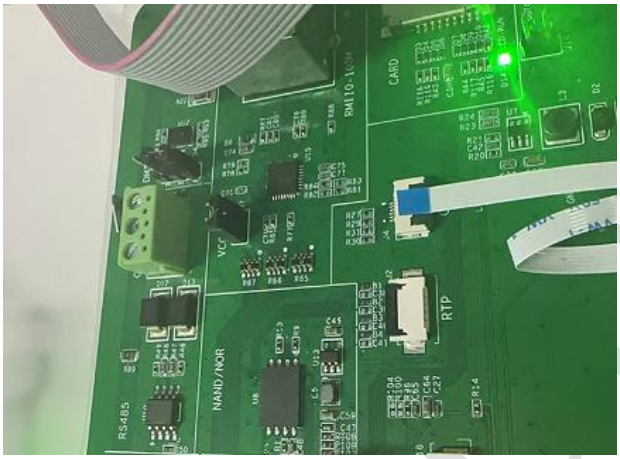
运行

```
scons --menuconfig
```

关闭 I2C、以及 Touch panel：

```
Board options  --->
    [] Using i2c3
Drivers options  --->
    Peripheral  --->
        Touch Panel Support  --->
            Gt911 touch panel options  --->
                [] Using touch panel gt911
```

2.若使用 JTAG 口，另外需断开 CTP 触屏排线：



8.3.2. VSCode 配置

上述 Dxx load 和 Dxx connect only 配置要使用起来前，都需要修改 Luban-Lite/.vscode/launch.json 文件中的部分参数，必须要和当前方案配置保持一致，主要修改：

1. 路径名、elf 文件名
2. DebugServer 的服务 IP 和端口号
3. 断点。

以 Dxx load 为例，修改方法如下：

```
{
    "name": "D12x/D13x Load",
    "type": "cppdbg",
    "request": "launch",
    "cwd": "${workspaceFolder}",
    "program": "${cwd}/output/d13x_demo88-nor_rt-thread_helloworld/images/d13x.elf",   // FIXME
    "args": [],
    "stopAtEntry": false,
    "environment": [],
    "externalConsole": true,
    "MIMode": "gdb",
    "miDebuggerPath": "${cwd}/toolchain/bin/riscv64-unknown-elf-gdb.exe",
    "setupCommands": [
        {
            "description": "Enable pretty-printing for gdb",
            "text": "-enable-pretty-printing",
            "ignoreFailures": true
        },
        {
            "text": "set arch riscv:rv32"
        },
        {
            "text": "set height 0"
        },
        {
            "text": "mem 0x30040000 0x3013ffff rw"
        },
        {
            "text": "mem 0x10000000 0x19ffffff rw"
        },
        {
            "text": "mem 0x40000000 0x41ffffff rw"
        },
        {
            "text": "target remote 172.16.11.126:1025"  // FIXME
        },
        {
            // MUST use full path
            "text": "load e:/AIC/luban-lite/luban-lite/output/d13x_demo88-nor_rt-thread_helloworld/images/d13x.elf" // FIXME
        },
        {
            // MUST use full path
            "text": "file e:/AIC/luban-lite/luban-lite/output/d13x_demo88-nor_rt-thread_helloworld/images/d13x.elf" // FIXME
        },
        {
            // FIXME
            "text": "b rt_hw_board_init"
        },
```

1.修改为当前方案的配置

2.T-HeadDebugServer正常链接JTAG调试器后，会打印可连接的IP和端口

3.修改为当前方案的配置
注意：这里必须是全路径

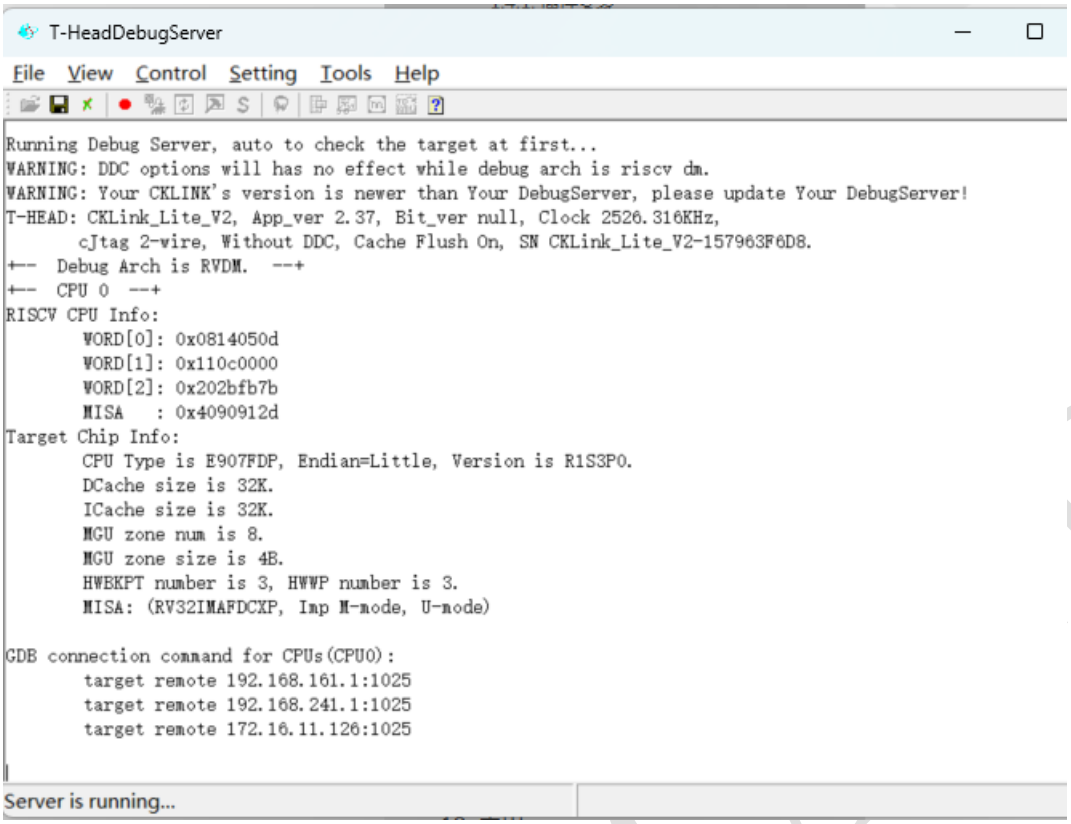
4.修改为自己的断点

Dxx connect only 需要修改的参数和上面类似，在 launch.json 文件中都用关键字 FIXME 标注。如果要添加多个断点，方法如下：

```
53              // FIXME
54              "text": "b rt_hw_board_init"
55          },
56          {
57              "text": "b aic_board_pinmux_init"
58          },
```

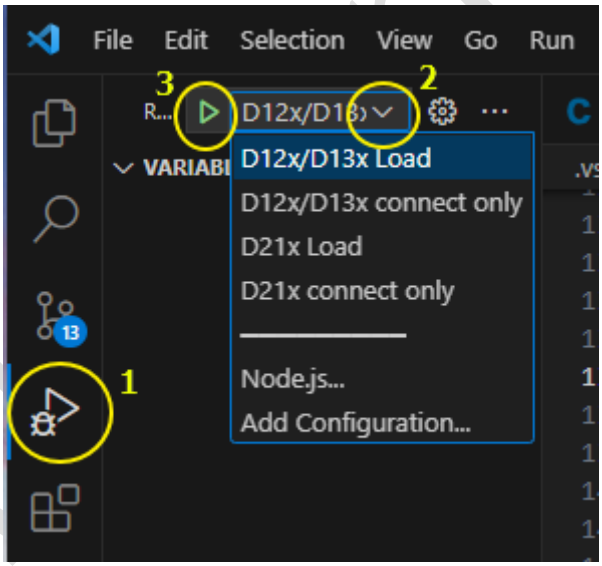
8.4. 进入调试

先打开 DebugServer，连接 JTAG 调试器成功后，界面如下：

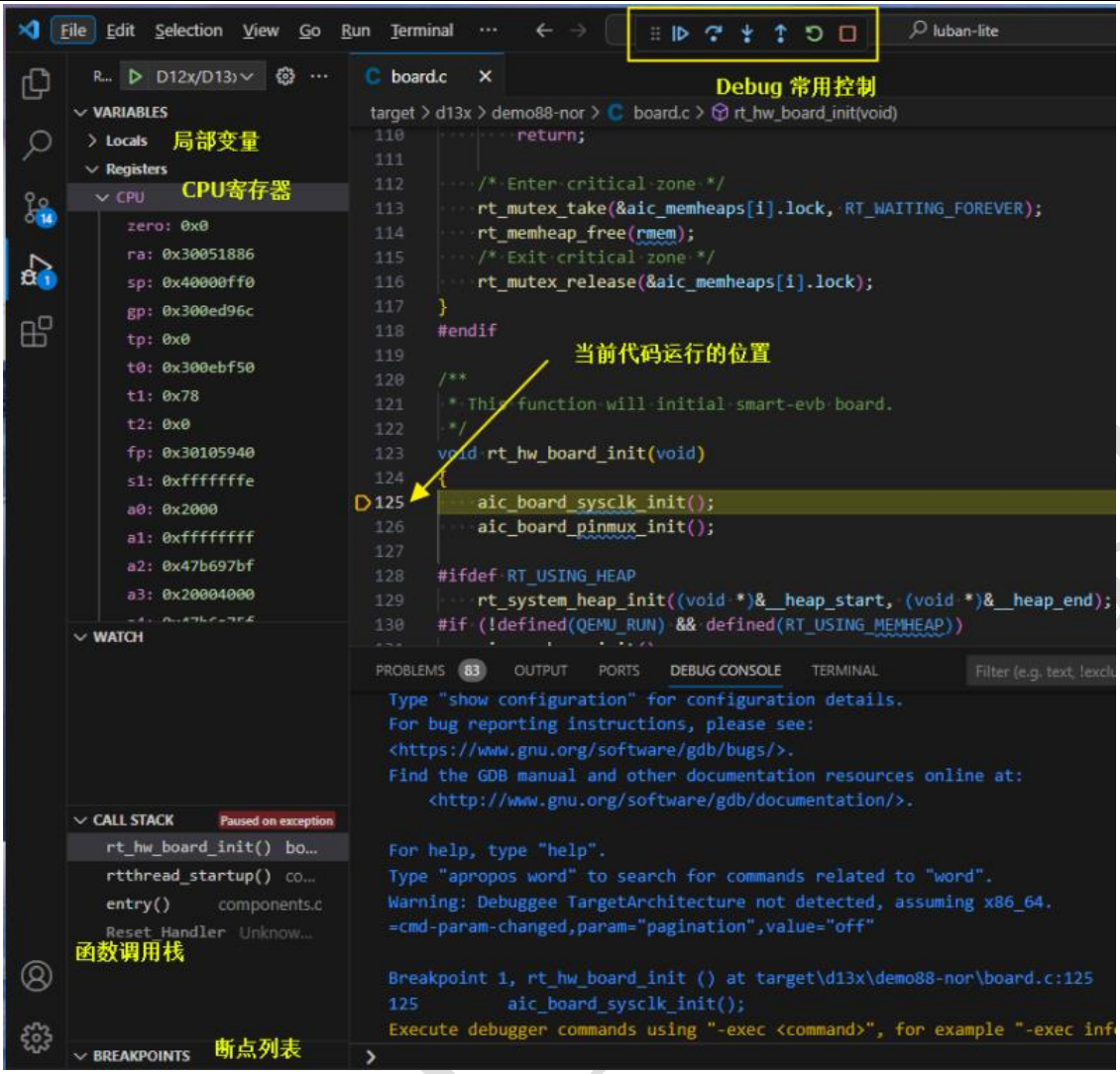


注: DebugServer 的服务端口修改方法： Setting -> Socket Setting

在 VSCode 中选择对应的 Debug 配置（详见：两种场景）：



以上操作顺利的话，VSCode 会进入 Debug 界面，如下：



接下来就可以进行通常的 Debug 调试了。

9. 文档资源

9.1. 文档中心

ArtInChip 文档中心可供用户在线查阅所有文档资源，官方网址：http://aicdoc.artinchip.com。

9.2. Gitee 下载

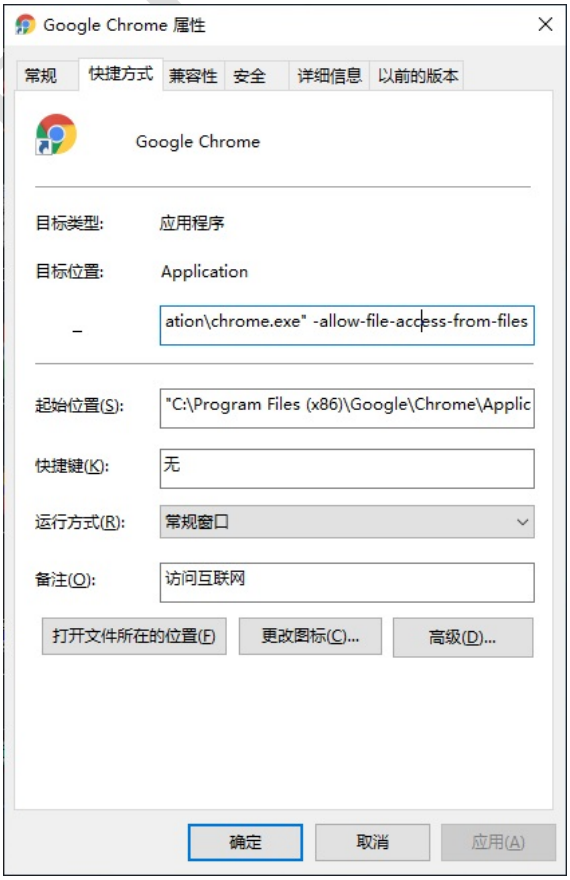
D12x 的相关文档使用 Gitee 存储和管理，也是开源仓库，可以通过下面的链接进行下载，也可以直接索取：

```
git clone https://gitee.com/artinchip/D12X-doc.git
```

9.3. 本地搜索

使用本文档如果希望搜索的内容更丰富，需要开启浏览器的本地文件访问权限，以 Chrome 为例：

- 打开 Chrome 应用的属性
- 选中"快捷方式"页
- 在"目标"后面加上 " –allow-file-access-from-files"，注意前面有个空格
- 重新打开 Chrome 即可



10. 教学视频

10.1. 教学视频链接

https://space.bilibili.com/3546578952390720?spm_id_from=333.1007.0.0