欢迎参加ST&利尔达联合举办的STM32线下实训

请大家拷贝U盘资料《ST-LIERDA_IC610高阶实战培训资料》到电脑，磁盘需大于100G 。

安装“软件”目录下工具（参考对应指导文档）：

1、SetupSTM32CubeMX-6.14.0-Win.exe

2、SetupSTM32CubeProgrammer_win64.exe

3、VMware-workstation-full-17.6.3-24583834.exe（已安装可跳过）

解压缩“虚拟机”目录下文件：

1、Ubuntu2404.7z

拷贝完成后，请将U盘传递给其他学员。

wifi:

SSID: 祥瑞亭-*

密码：85269888

进入STM32MPU技术交流群，

培训中将发送相关资料

STM32公众号

利尔达公众号

让万物互联更简单!

演讲主题	时间段
签到和准备	8:30-9:30
ST 及 MP25 介绍	9:30-9:40
利尔达及 IC610 介绍	9:40-9:50
IC610 开发板资源介绍	9：50-10：00
STM32MP25xSDK 介绍	10：00-10：10
Linux 环境搭建及代码编译	10：10-10：40
IC610 网络模块移植	10：40-11：40
午休	
IC610 MIPI DSI 显示模块移植	13：30-13：50
IC610 EVK 镜像烧录及启动	13：50-14：00
IC610 TSN 案例分享及演示	14：00-15：00
中场休息，问卷时间	15：00-15：15
IC610 MIPI CSI 摄像头模块移植	15：15-15：45
3 个示波器抽奖	15：45-15：50
IC610 AI 案例分享及演示	15：50-16：50
5 套开发板抽奖	16：50-16：55
颁奖大合影	16：55-17：00

产品名称：ST-A35-IC610 工业核心板

产品型号：L-IDMIM0-AA185

版本：Rev1.1

公司地址：杭州市文一西路 1326 号利尔达物联网科技园　Tel：0571-88800000

法律声明

若接收利尔达科技集团股份有限公司(以下称为“利尔达”)的此份文档，即表示您已经同意以下条款。若不同意以下条款，请停止使用本文档。

本产品符合有关环境保护和人身安全方面的设计要求，产品的存放、使用和弃置应遵照产品手册、相关合同或者相关法律、法规的要求进行。

本公司保留在不预先通知的情况下，对此手册中描述的产品进行修改和改进的权利；同时保留随时修订或收回本手册的权利。

文件修订历史

文档版本	变更日期	修订人	审核人	变更内容
Rev1.0	25-03-15	YQA		初始版本
Rev1.1	25-10-5	YQA		优化源码编译步骤，增加 tf-a 编译错误提示

目录

1 引言

本文档依托 IC610 evk，旨在搭建 A35 linux SDK，包括 Ubuntu 环境搭建、烧录镜像制作、烧录工具安装、镜像烧录、tsn 开发环境搭建、tsn 功能测试、ai 环境搭建及日常 demo 测试。

2 开发资源准备

2.1 资源的准备

软件 list:

1、PC 电脑（Windows10 及以上系统），

2、软件 VMWare17（需自行安装，培训资料夹下虚拟机\VMware-workstation-full-17.6.3-24583834.exe），

3、Xshell 或 SecureCRT

4、Ubuntu2404 虚拟机，可使用 U 盘提供的虚拟机（需自行配置和启动虚拟机，U 盘位置：ST-LIERDA_IC610 高阶实战培训资料\虚拟机\ubuntu2404\Ubuntu 64 位.vmdk），安装参考文档 doc\《vmware 及 ubuntu2404 安装》 。

5、咨询资料夹软件目录下安装: CH343SER.EXE、SetupSTM32CubeMX-6.14.0-Win.exe、SetupSTM32CubeProgrammer_win64.exe

ST-LIERDA_IC610 高阶实战培训资料介绍:

doc：cpu 相关手册及 ic610 软件开发相关指导;

Image：IC610 evk 固件镜像

src-cubemx：ic610 evk 的 cubemx 源码及 STM32Cube_FW_MP2

Video：效果视频

软件：windows 下软件包及相关驱动包

虚拟机：Ubuntu2404 镜像包

硬件 list：

1、2 个 IC610 开发板

2、2 根 USB TYPE-C 数据线，用于 debug 调试

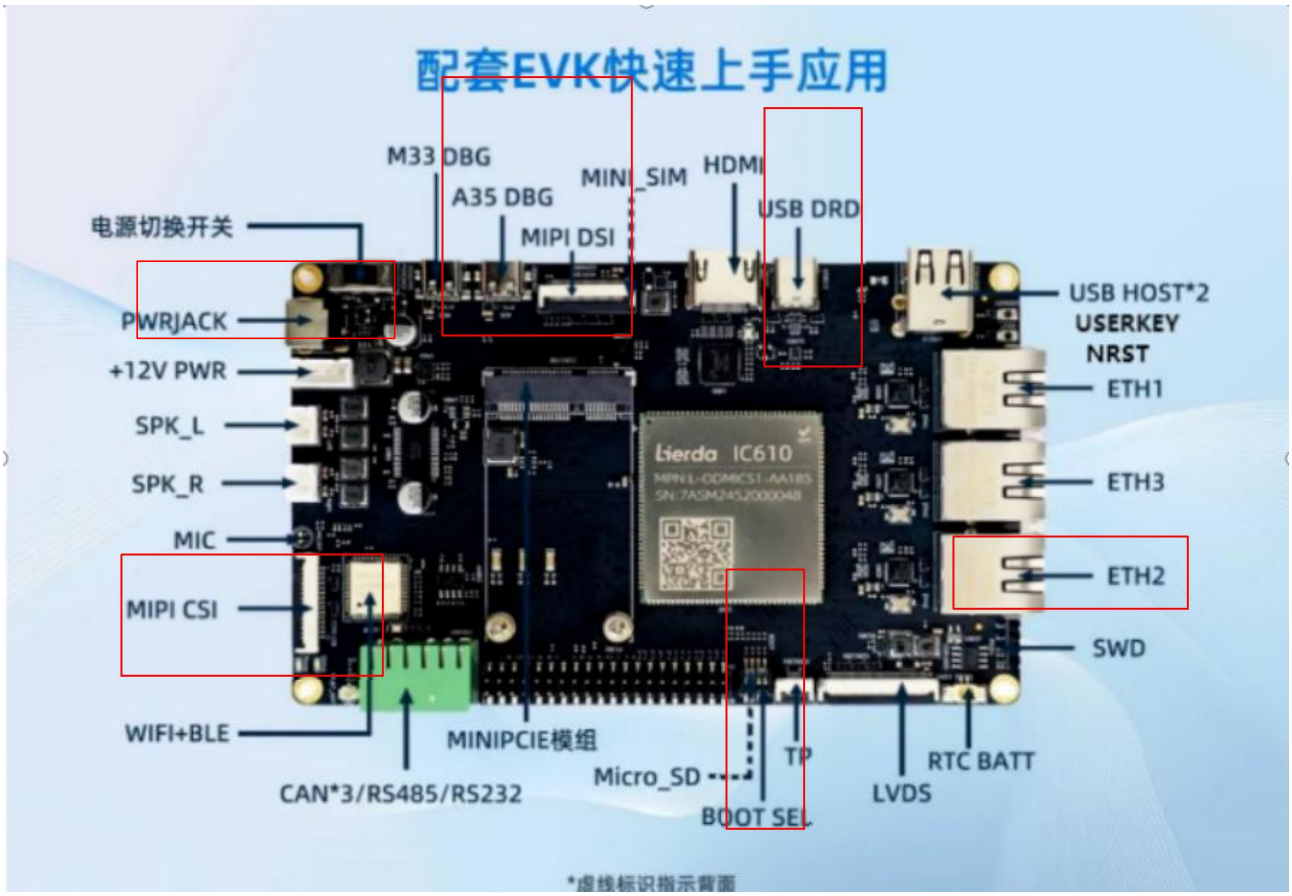
3、2 块 MIPI 显示模块

4、1 块 OV5640 数字摄像头模块

5、2 个 12V2A 电源

6、1 根千兆以太网网线

2.2 IC610 开发板



注意事项

开发板

烦请大家在今天活动结束后，务必将开发板归还至前台并且在返还表中对应项签名，我们需要流转至下一场培训。

如果中途需要离场，也请先将开发板等归还至前台，谢谢大家的配合。

午餐

中午为大家准备了午餐，请大家凭餐券用餐

奖品

示波器、开发套件和淘宝专属优惠券

问卷

感谢您的参与，
我们期待听到您的反馈！

2.2.1 硬件参数

表 1-1 硬件参数

序号	项目	参数说明	备注
1	处理器	STM32MP255DAK3	双核 A35+M33
2	内存	1024MB	DDR4
3	Flash 存储器	8GB	eMMC
4	电源接口	DC 12-28V / 2A	防反接，低压、过压保护
5	指示灯	2 个	DCIN、heartbeat
6	以太网	3 个	千兆以太网，eth3 使用需要使用 cpu257
7	SIM 卡插槽	1 个	MiniSIm
8	USB HOST	1 个	Usb2.0 Type-A

9	USB OTG	1 个	Type-C
10	Mini PCI-e	1 个	和 3G/4G 模块同一个接口
11	RTC	2 个	cpu 自带 1 个，硬件 rtc 1 个
12	看门狗	1 个	cpu 自带
13	MicroSD 插槽	1 个	内部板载
14	A35 调试口	1 个	Type-C
15	LVDS	1 个	1280*800
16	RS232	1 路	插针
17	CAN	3 路	插针
18	RS485	1 路	插针
19	MIPI	1 路	800*1280
20	KEY	2 个	Reset 复位按键、用户按键
21	HDMI	1 路	自适应
22	Camera	1 路	ov5640、 ov5647
23	触摸	1 路	Lvds 和 mipi 共用 1 路，不可同时使用
24	M33 调试口	1 路	Type-C
25	Audio	1 路	AUX 接口
26	WiFi	1 路	SDIO 接口
27	蓝牙	1 路	Uart 接口
28	树莓派插针	1 路	Spi*2 、i2c、uart、gpio*3

2.2.2 软件参数

表 1-2 软件参数

类别	版本	描述
TF-A	tf-a-stm32mp-v2.10.5	保护设备启动过程的安全
OPTEE	optee-os-stm32mp-4.0.0	硬件隔离保护敏感数据，防止非安全世界的访问
Bootloader	u-boot-2023.10	系统引导程序，负责系统初始化和引导内核启动
Kernel	Linux-6.6.48	ST 官方内核版本
文件系统	ext4	基于 st 官方 yocto 定制的文件系统，支持 SD 启动更新 emmc 固件

2.2.3 Boot

本产品默认存储为 emmc，sd 卡启动可用于生产测试或镜像烧录等功能。

引脚实物图，拨码 1234 分别对应 boot0-boot3。

启动方式	Boot0 boot1 boot2 boot3
Emmc	off on off off
Sd 卡	on off off off
Otg 烧录	off off off off

Sd 卡启动

Emmc 启动：

OTG 烧录模式：

emmc 或 sd 卡启动模式下，上电后，按住 USERKEY，自动进入烧录模式。

2.2.4 供电

适配器 DC 插头， 12-28V / 2A

2.2.5 系统登录

板子内使用的 usb 转串口芯片为 CH340G，pc 端需要安装该芯片驱动 CH343SER.EXE。

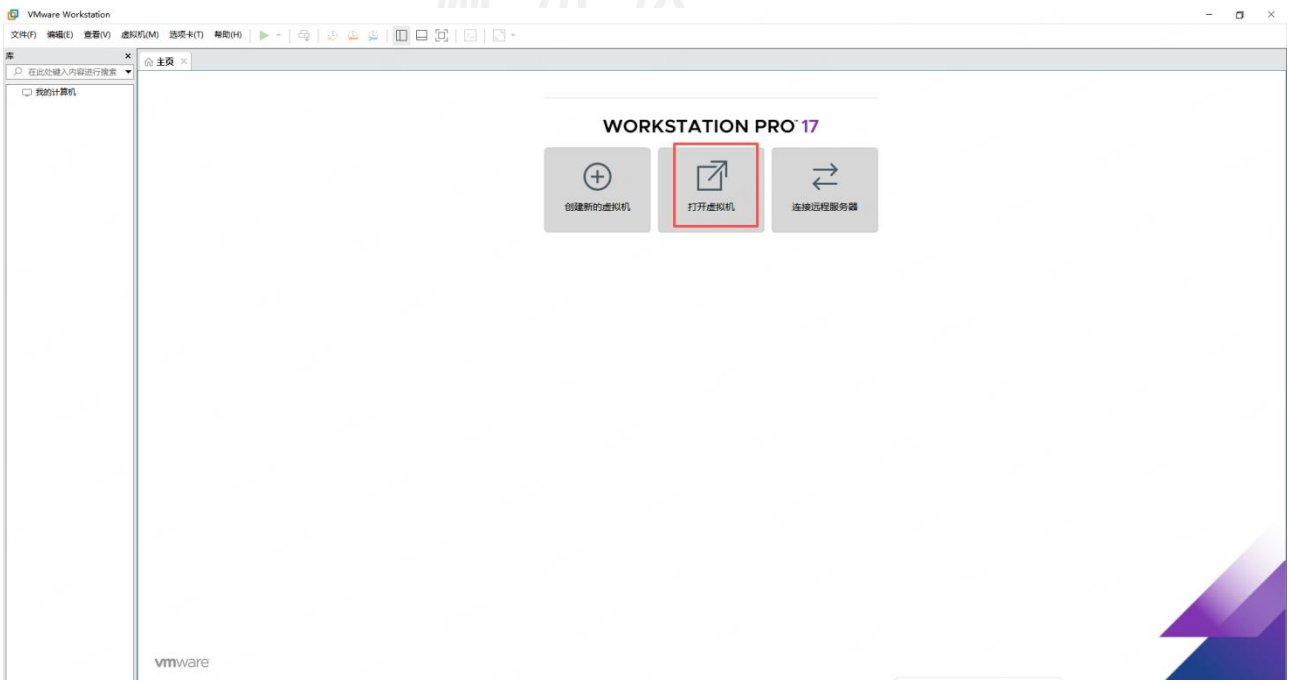
使用 typec 接口的 usb 线连接 A35 DBG 接口（如下），使用串口工具打开对应 com 口，端口查询设置参数：波波特 115200 、数据位 8、奇偶位 0、停止位 1、流控无，如下。

3 开发环境搭建及 SDK 编译

本版本 yocto 编译需要 ubuntu2404 以上，内存 16G+,存储可用空间 150GB+。

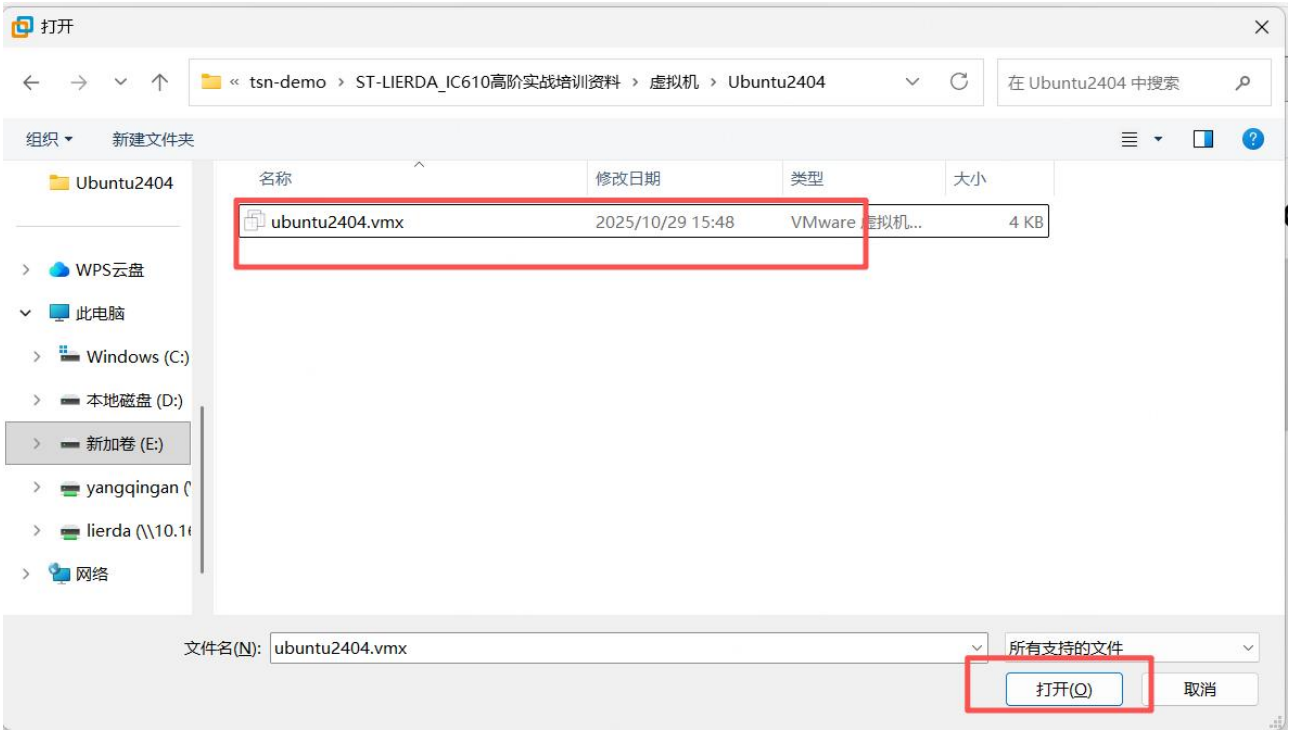
零基础安装可参考安装文档 doc\《vmware 及 ubuntu2404 安装》，本次培训可直接使用 vmware 打开 ST-LIERDA_IC610 高阶实战培训资料\虚拟机\ubuntu2404\ubuntu2404.vmx。若本机未安装 vmware 可参考 doc\《vmware 及 ubuntu2404 安装》1.1 vmware 安装。

Vmware 打开后，点击“打开虚拟机”

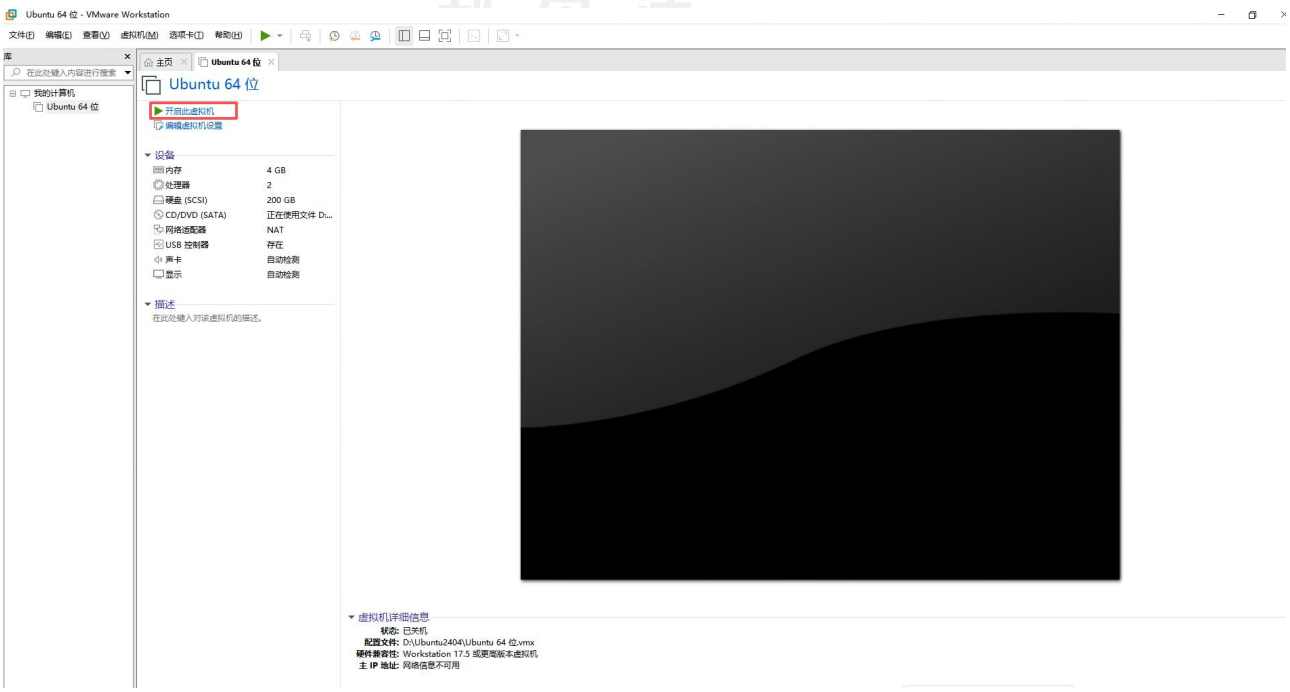


找到 ST-LIERDA_IC610 高阶实战培训资料\虚拟机\ubuntu2404\ubuntu2404.vmx，选

择打开即可。



在 PC 的 E 盘下创建空文件夹名称为“vmware”，用于共享文件，无此文件夹，ubuntu 可能无法启动。



点击“开启此虚拟机”系统自动启动。

此虚拟机用户名为 lsd，密码为 lsd123。

3.1 A35 SDK 编译

3.1.1 tf-a 编译

/home/lsd/ic610/src-ic610 为 sdk 源码路径

```
cd /home/lsd/ic610/src-ic610
```

tf-a 编译

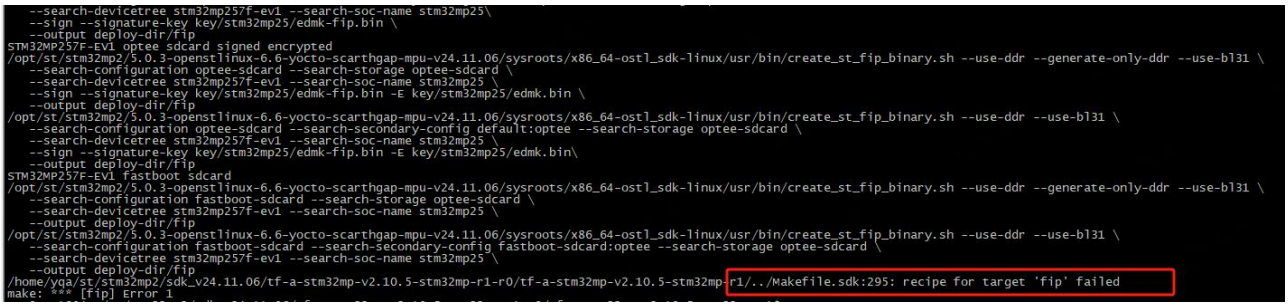
```
src-ic610$ cd tf-a-stm32mp-v2.10.5-stm32mp-r1-r0/tf-a-stm32mp-v2.10.5-stm32mp-r1/
```

编译：

```
$: ./build.sh
```

注意：

首次编译报错退出，是正常现象，继续编译 optee、u-boot 即可。



编译完成后

```
$: ls ../../ic610-images/arm-trusted-firmware/
```

metadata.bin tf-a-stm32mp255d-ic610-mx-optee-emmc.stm32 tf-a-stm32mp255d-ic610-mx-optee-sdcard.stm32 tf-a-stm32mp255d-ic610-mx-usb.stm32

3.1.2 Optee 编译

```
src-ic610$ cd optee-os-stm32mp-4.0.0-stm32mp-r1-r0/optee-os-stm32mp-4.0.0-stm32mp-r1/
```

编译脚本：

```
$: ./build.sh
```

编译完成后生成

```
$: ls ../../ic610-images/optee/
```

tee-header_v2-stm32mp255d-ic610-mx.bin tee-pageable_v2-sstm32mp255d-ic610-mx.bin tee-pager_v2-stm32mp255d-ic610-mx.bin

tee-header_v2-stm32mp255d-ic610-mx.bin tee-pageable_v2-stm32mp255d-ic610-mx.bin tee-pager_v2-stm32mp255d-ic610-mx.bin

3.1.3 u-boot 编译

```
src-ic610$ cd u-boot-stm32mp-v2023.10-stm32mp-r1-r0/u-boot-stm32mp-v2023.10-stm32mp-r1
```

编译：

```
$: ./build.sh
```

编译完成后生成

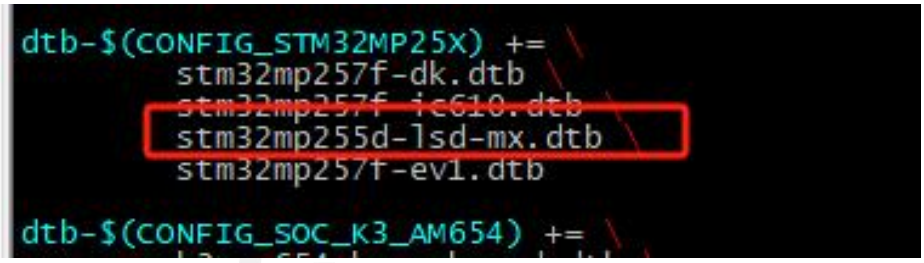
```
$: ls ../../ic610-images/fip/
```

fip-stm32mp255d-ic610-mx-ddr-optee-emmc.bin fip-stm32mp255d-ic610-mx-ddr-optee-sdcard.bin fip-stm32mp255d-ic610-mx-optee-emmc.bin fip-stm32mp255d-ic610-mx-optee-sdcard.bin

注意：stm32mp255d-ic610-mx 为要编译的 dts 名称，编译其他 dts 修改该参数即可；同时需要在 arch/arm/dts/Makefile dtb-$(CONFIG_STM32MP25X) += \，添加对应 dts。

如新 dts 为 stm32mp255d-lsd-mx

```
vim arch/arm/dts/Makefile
```



3.1.4 Kernel 编译

```
src-ic610$: cd linux-6.6.48/linux-6.6.48/
```

编译命令

```
$: ./build.sh
```

编译完成后镜像为

```
$: ../../ic610-images/bootfs/
```

Image.gz stm32mp255d-ic610-mx.dtb stm32mp257f-dk.dtb stm32mp257f-ev1.dtb

镜像自动更新到烧录镜像 st-image-bootfs-openstlinux-weston-stm32mp2.ext4

该 build.sh 脚本自动编译 kernel、wifi 模组驱动、gpu 驱动及更新文件系统等，用户可

自行修改简化该脚本。

Kernel 配置：

```
source /opt/st/stm32mp2/5.0.3-openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06/environment-setup-cortexa35-ostl-linux
make menuconfig
```

Kernel 单独编译 dtb：

```
make dtbs
```

编译完成后 log 输出重新编译的 dtb 镜像，

如 arch/arm64/boot/dts/st/stm32mp255d-ic610-mx.dtb 。

Kernel 单独编译 Image.gz：

```
make Image.gz vmlinux LOADADDR=0xC2000040
```

编译完成后生成：arch/arm64/boot/Image.gz

注意：

单独编译后镜像未自动打包到

ic610-images/st-image-bootfs-openstlinux-weston-stm32mp2.ext4 下，需要手动打包镜像。

source

/opt/st/stm32mp2/5.0.3-openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06/environment-setup-cortexa35-ostl-linux

为使能交叉编译器环境，在同一个终端下执行一次即可，若打开新的终端需要重新执行。

3.1.5 自动编译

src-ic610 下 build.sh 为自动编译全部镜像脚本，运行 build.sh 自动编译全部镜像，单步编译可参考软件用户手册。

```
src-ic610$ ./build.sh
```

注意：build.sh 默认编译的 dts 为 stm32mp255d-ic610-mx，编译其他 dts 需要修改 build.sh 内变量 dts 或增加 dts 参数，以下无需还行，如

编译完成后生成全部镜像：

```
ic610-images/FlashLayout_sdcard_stm32mp255d-ic610-mx-optee.raw
ic610-images/fip/
fip-stm32mp255d-ic610-mx-ddr-optee-emmc.bin fip-stm32mp255d-ic610-mx-optee-emmc.bin
fip-stm32mp255d-ic610-mx-ddr-optee-sdcard.bin fip-stm32mp255d-ic610-mx-optee-sdcard.bin

ic610-images/arm-trusted-firmware/
metadata.bin tf-a-stm32mp255d-ic610-mx-optee-emmc.stm32 tf-a-stm32mp255d-ic610-mx-optee-sdcard.stm32 tf-a-stm32mp255d-ic610-mx-usb.stm32

st-image-bootfs-openstlinux-weston-stm32mp2.ext4
```

st-image-vendorfs-openstlinux-weston-stm32mp2.ext4

st-image-weston-openstlinux-weston-stm32mp2.ext4

3.2 ic610-images 简介

目录	功能	备注
arm-trusted-firmware	tf-a 镜像目录	源码编译镜像自动更新至此
optee	optee 镜像目录	源码编译镜像自动更新至此
fip	u-boot 镜像目录	源码编译镜像自动更新至此
bootfs	Kernel 镜像目录	源码编译镜像自动更新至此
flashlayout_st-image-westo n	系统工具，不可修改及删除	
scripts	系统工具，不可修改及删除	
make_raw.sh	制作 sd 卡烧录镜像工具	脚本内参数需要根据实际修改： sd_dev=/dev/sdb dts=stm32mp255d-ic610-mx
emmc_burn	Sd 卡烧录 emmc 镜像包	emmc_install_command.sh 根据实际问修改
st-image-bootfs-openstlinu	烧录 bootfs 镜像	bootfs 目录镜像更新到 ext4 内

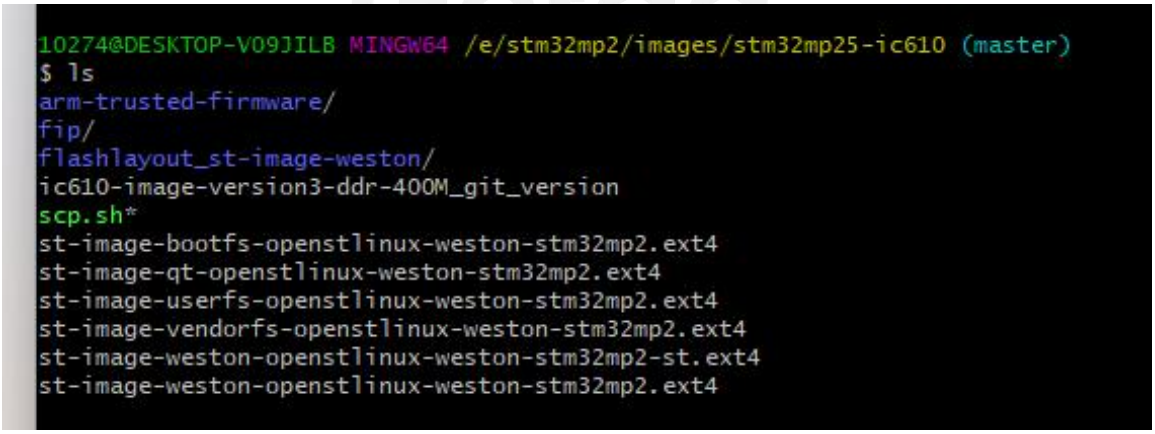
x-weston-stm32mp2.ext4		
st-image-userfs-openstlinux-weston-stm32mp2.ext4	烧录 userfs 镜像	
st-image-vendorfs-openstlinux-weston-stm32mp2.ext4	烧录 wendorfs 镜像	
st-image-weston-openstlinux-weston-stm32mp2.ext4	烧录 weston 镜像	
rootfs	Weston和userfs镜像自定义	可对 weston 镜像和 userfs 镜像修改及添加应用层等

3.2.1 烧录镜像更新到 windows

Windows 下下载 git 并安装，下载路径 https://gitforwindows.org/ 。

ic610-sdk\images\stm32mp25-ic610-rev1.rar 为镜像烧录包，解压后进入。

安装 git 后在镜像烧录文件夹内击桌面选择 Git Bash Here，创建 shell 窗口。



修改 scp.sh 内变量

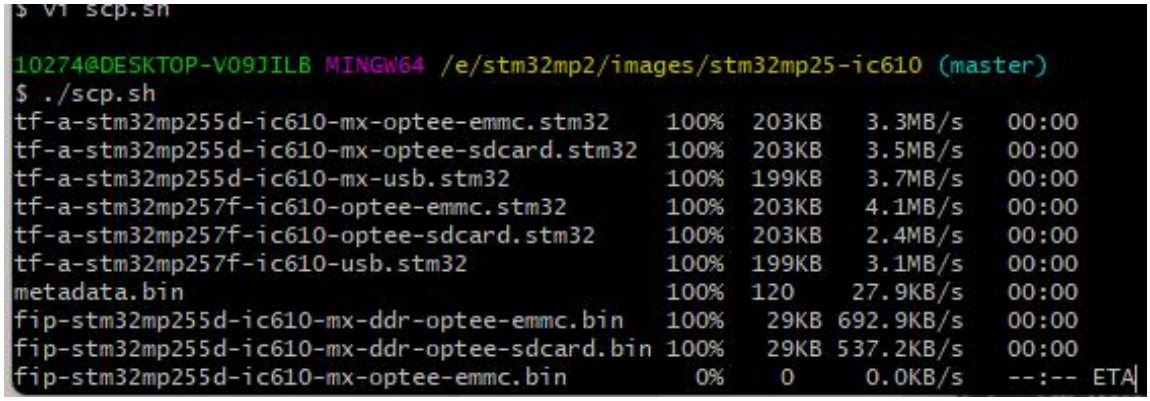
```
path=/home/yqa/st/stm32mp2/sdk_v24.11.06

ubuntu_ip=192.168.2.103

user=yqa
```

path 为 sdk 源码路径，ubuntu_ip 为 ubuntu 开发环境 ip（开发环境和 windows 在同

一个局域网内）

user 为 ubuntu 的用户名。

修改完成后运行./scp.sh 将镜像更新到烧录包内。



镜像烧录参考 4.2 镜像烧录。

3.2.2 自定义文件系统

SDK 下 ic610-images/rootfs 内为文件系统 st-image-weston-openstlinux-weston-stm32mp2.rootfs.ext4 和 st-image-userfs-openstlinux-weston-stm32mp2.userfs.ext4 完善脚本，用户可修改 perfect.sh 添加相关内容。

开机脚本为 ic610-images/rootfs 下 etc/rc.local

文件系统若在 yocto 下重新编译完善后，可直接替换 ic610-images 下的 st-image-weston-openstlinux-weston-stm32mp2.ext4，然后删除 ic610-images/rootfs/fs 目录，

```
ic610-images/rootfs$ sudo ./perfect-fs.sh
```

则将更新后的文件系统添加用户修改后重新打包至 ic610-images 下。

3.2.3 sd 卡启动卡镜像 raw 制作

注意：此镜像用于 sd 卡启动及 sd 卡启动后烧录 emmc，raw 镜像制作过程中自动将

emmc 烧录镜像打包至镜像中。

make_raw.sh 为制作 raw 镜像脚本

```
ic610-images$ sudo ./make_raw.sh
```

raw 镜像 FlashLayout_sdcard_stm32mp255d-ic610-optee.raw ，该镜像需要自 Ubuntu 下手动烧录，或复制到 windows 下参考 4.3 raw 镜像烧写。

make_raw.sh 参数介绍：

dts=stm32mp255d-ic610-mx 为设备树名称，需要根据实际 dts 进行修改，目前支持 stm32mp257f-ic610 和 stm32mp255d-ic610-mx 其他 dts 用户可自行修改并增加适配的 tsv 文件。

脚本依赖的 tsv 文件如下：

flashlayout_st-image-weston/optee/FlashLayout_sdcard_stm32mp255d-ic610-mx-optee.tsv

flashlayout_st-image-weston/optee/FlashLayout_sdcard_stm32mp255d-ic610-mx-optee.tsv

烧录 raw 镜像到 sd 卡种，/dev/sdb 为 sd 卡节点，具体以实际节点为准，防止破坏虚拟机环境：

```
ic610-images$ sudo dd if=./FlashLayout_sdcard_stm32mp255d-ic610-optee.raw of=/dev/sdb bs=8M conv=fdatasync status=progress
```

Window 是下制作烧录启动卡可参考 doc/Lierda_IC610_EVK 软件应用开发指导_Rev1.1.pdf

4 IC610 网络模块移植

4.1 Cubemx 使用

4.1.1 Cubemx 功能

STM32MP2 CubeMX 的主要作用可以分为以下几个层面：

1. 图形化引脚分配与冲突解决

作用：STM32MP2 芯片有大量功能丰富且复用的引脚。在 CubeMX 中，你可以通过拖拽的方式直观地分配每个引脚的功能（如 UART、I2C、GPIO 等）。

优势：工具会实时检查引脚冲突（例如，两个外设功能被分配到同一个引脚），并用颜色高亮显示，有效避免了硬件设计阶段的低级错误。

2. 时钟树配置

作用：MPU 的时钟系统非常复杂，包括 PLL、分频器、多路选择器等，需要为 A 核、M 核、各种总线、外设提供不同频率的时钟。

优势：CubeMX 提供了可视化的时钟树图。你只需输入想要的频率（例如，配置 A35 核心跑 1.5GHz），工具会自动计算并设置所有相关的 PLL 和分频器参数，并验证配置的可行性。手动计算这些寄存器值极其繁琐且容易出错。

3. 外设与中间件配置

作用：对于每个外设（如 UART, I2C, SPI, Ethernet, GPU 等）和中间件（如 Linux 下的设备树配置），CubeMX 提供了详细的配置界面。你可以设置工作模式、参数（如波特率、地址等），而无需直接阅读数百页的参考手册。

优势：自动化生成初始化代码。配置完成后，CubeMX 可以生成对应外设的 C 初始化代

码（对于 Cortex-M 核）和 Linux 设备树源文件（.dts）（对于 Cortex-A 核）。

4. 项目管理与代码生成

作用：你可以指定项目名称、使用的 IDE（如 STM32CubeIDE, Keil, IAR）、工程路径等。

优势：

对于 Cortex-M 核：生成一个完整的、可直接编译和运行的嵌入式项目，包含所有外设的 HAL 库初始化代码和 main 函数框架。

对于 Cortex-A 核：生成 Linux 设备树，这是 Linux 内核识别和管理硬件资源的关键文件。开发者无需手动编写复杂的设备树，大大降低了入门门槛和出错概率。

5. 电源管理配置

作用：STM32MP2 具有复杂的电源管理单元，支持多种低功耗模式。

优势：CubeMX 可以帮助你配置这些电源模式，确保不同电源域的开/关序列正确，并生成相应的配置代码。

Cubemx 官方下载链接：https://www.st.com/en/development-tools/stm32cubemx.html

Cubemx 当前最新版本为 6.15.1，不同版本创建的 project 存在不兼容，故版本选择时必须与本开发板选择的版本一致。

IC610 使用 cubemx 构建工程及源码，cubemx 版本 6.14.0。

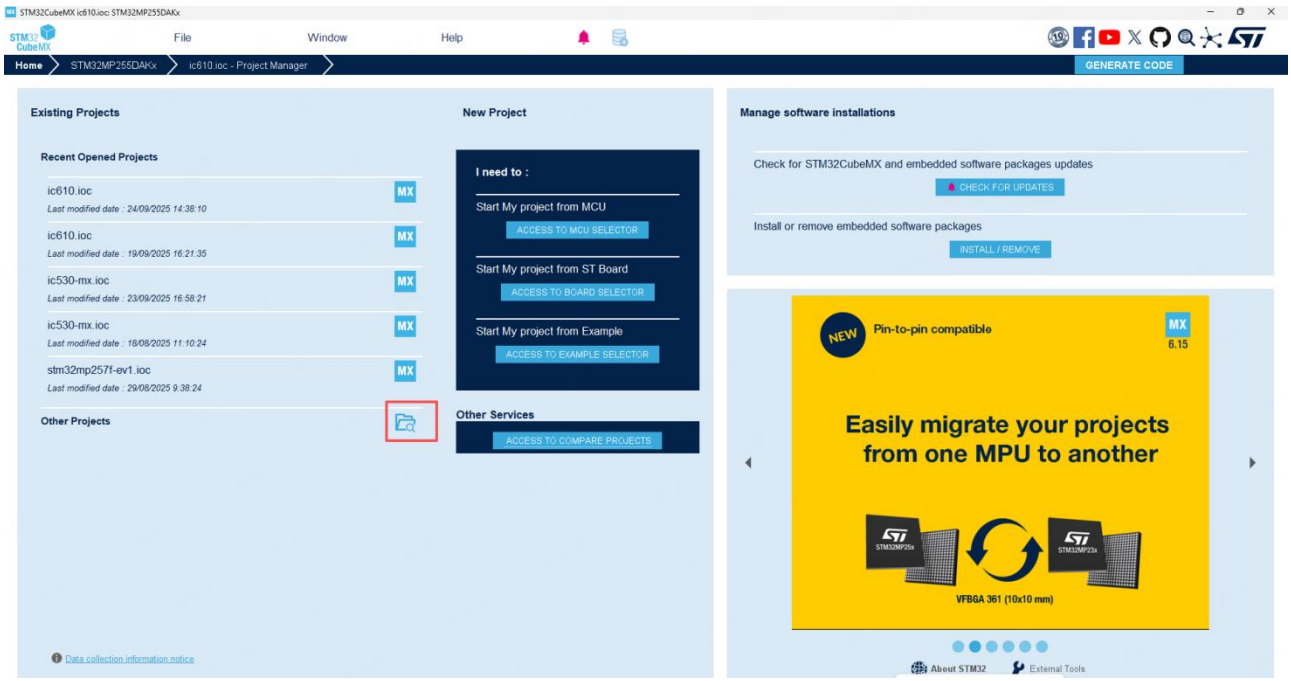
参考 doc\《Lierda_IC530&IC610_ST_cubemx 应用指导_Rev1.1》2.2 Cubemx 安装 安装 ST-LIERDA_IC610 高阶实战培训资料\软件\下 SetupSTM32CubeMX-6.14.0-Win.exe

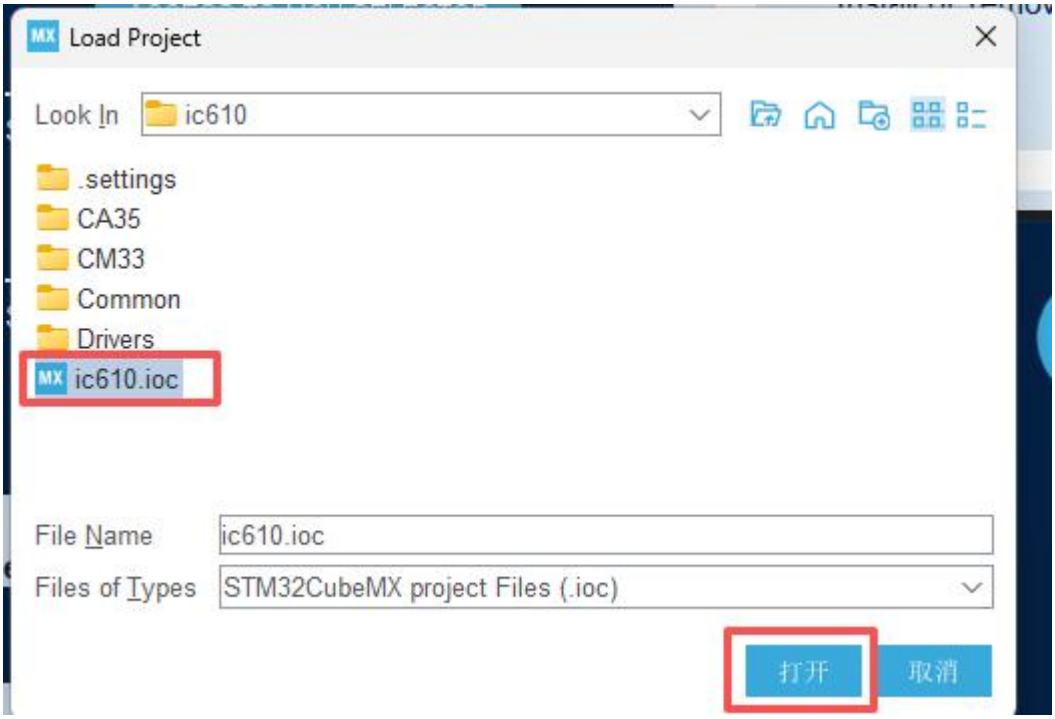
4.1.2 Cubemx 开发 ic610

IC610 cubemx 源码：路径为 ST-LIERDA_IC610 高阶实战培训资料\src-cubemx\ic610

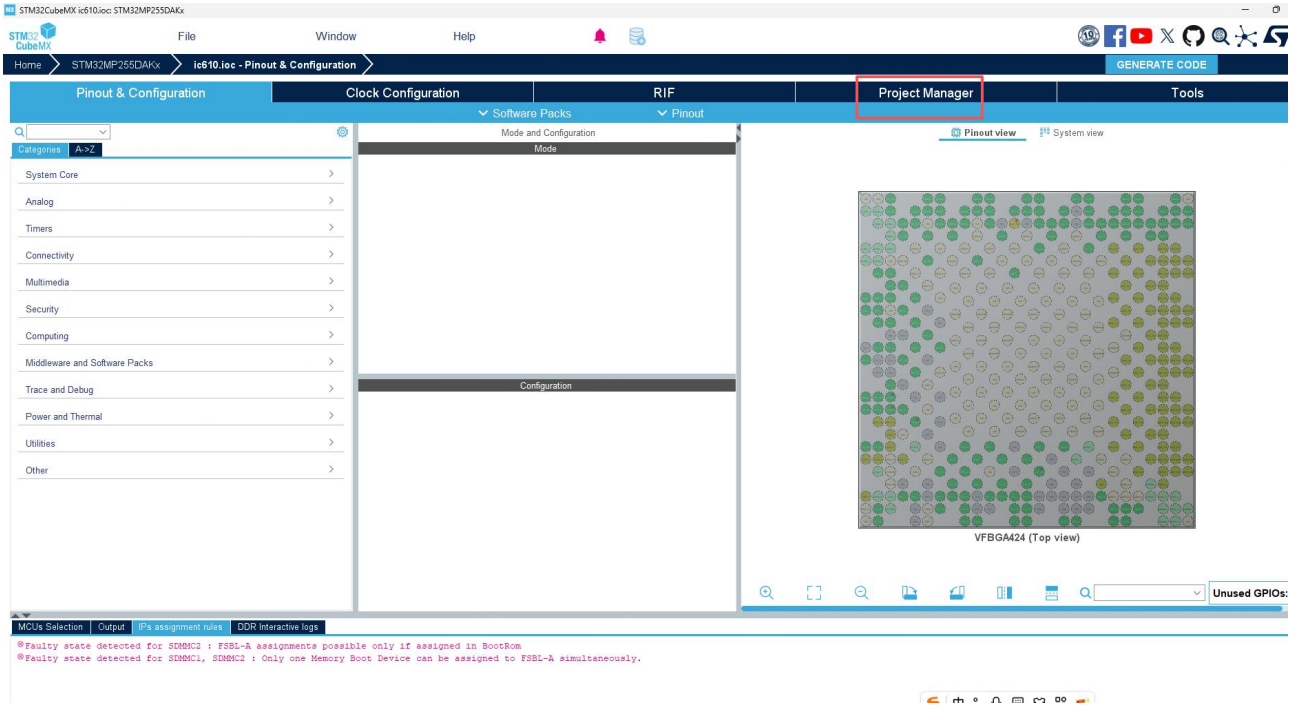
Firmware ： ST-LIERDA_IC610 高 阶 实 战 培 训 资 料 \src-cubemx\STM32Cube_FW_MP2_V1.1.0\

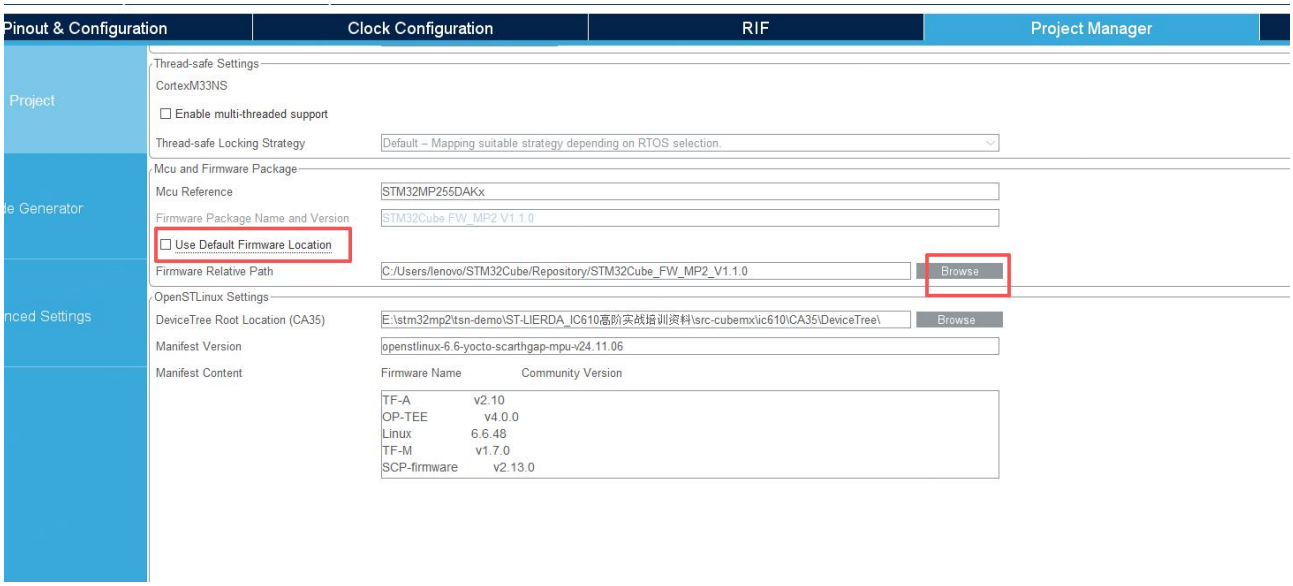
将上述源码和 firmware 解压后，cubemx6.14 打开 ic610.ioc。





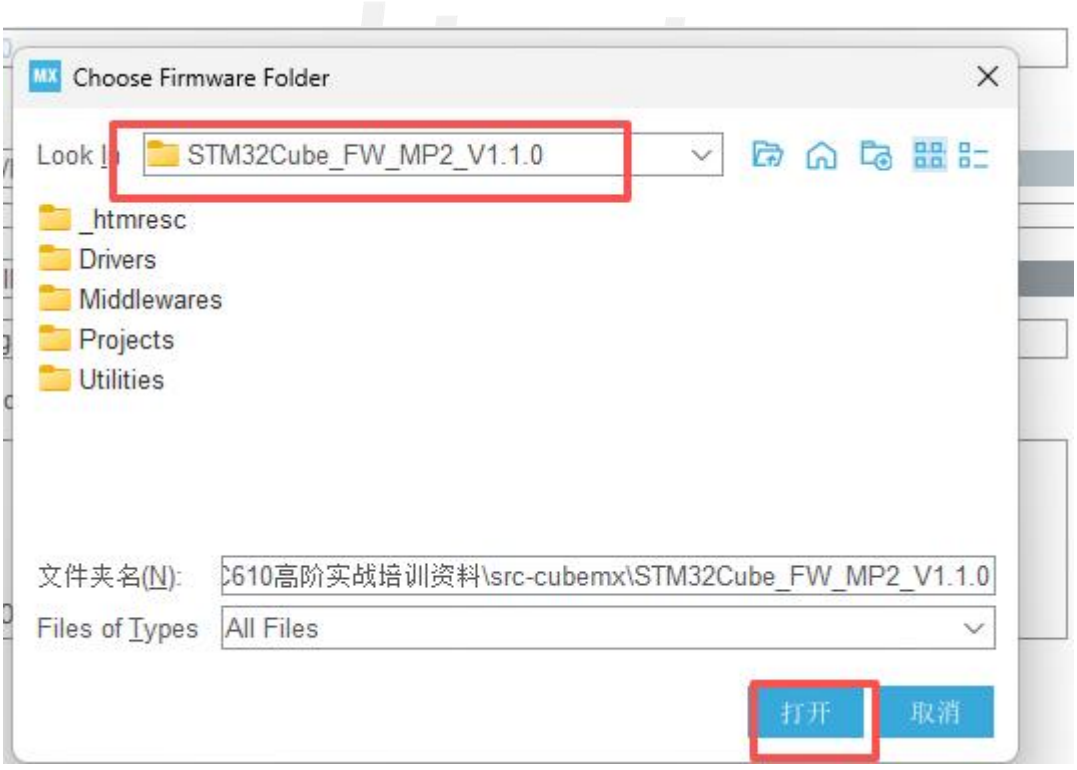
打开后界面如下，点击 Project Manger 设置 firmware

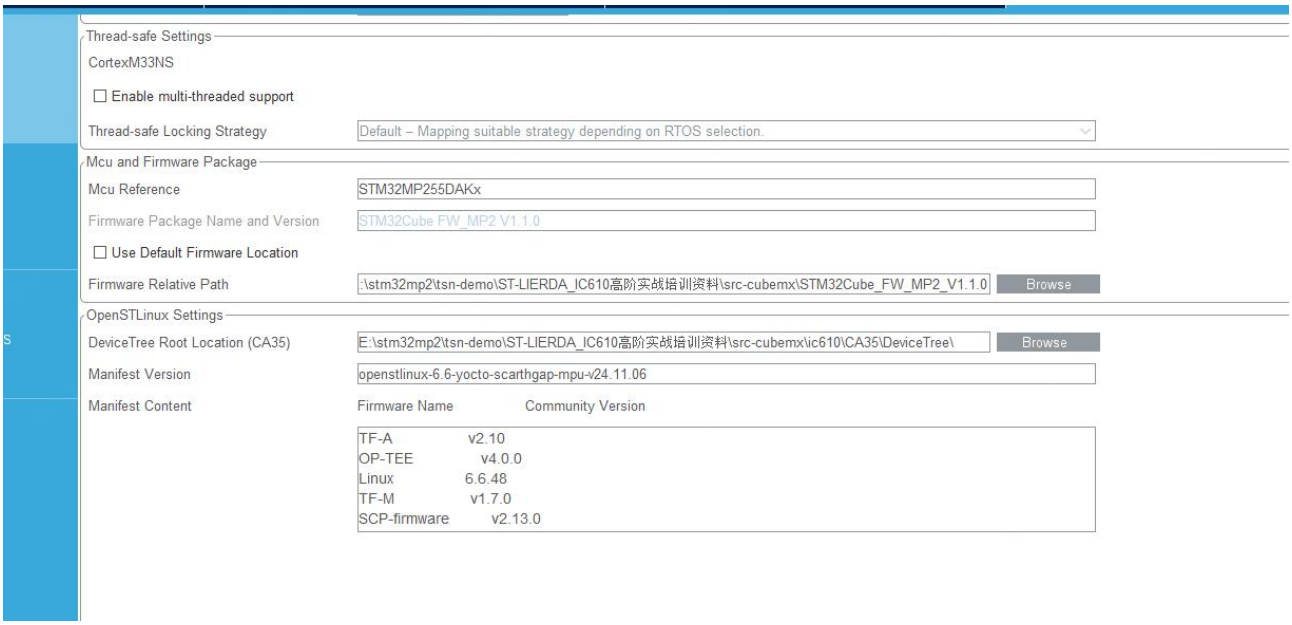




关闭 “Use Default Firmware Location”点击 Browse 设置为 ST-LIERDA_IC610 高阶实战培训资料\src-cubemx\STM32Cube_FW_MP2_V1.1.0\

设置如下：

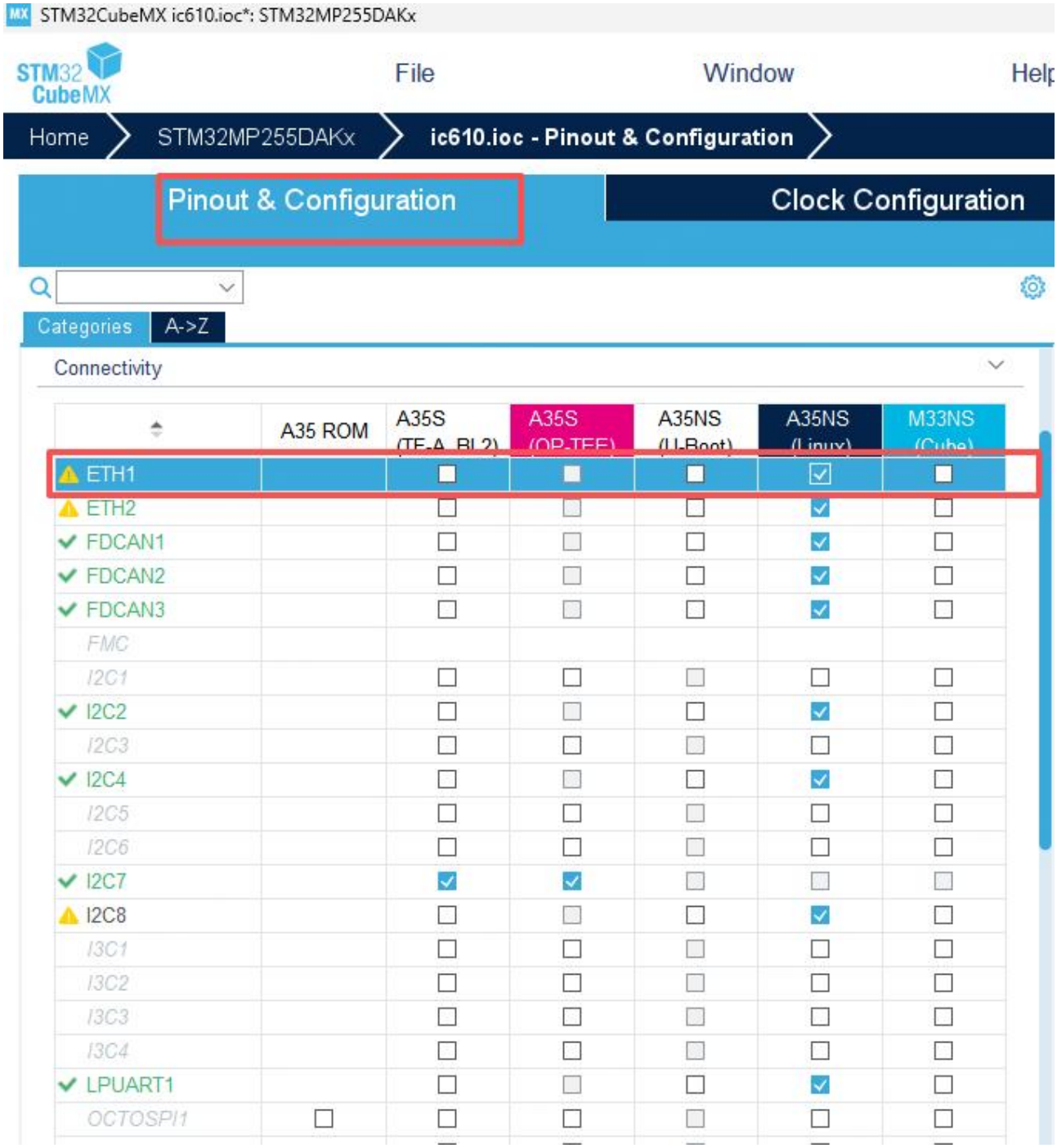




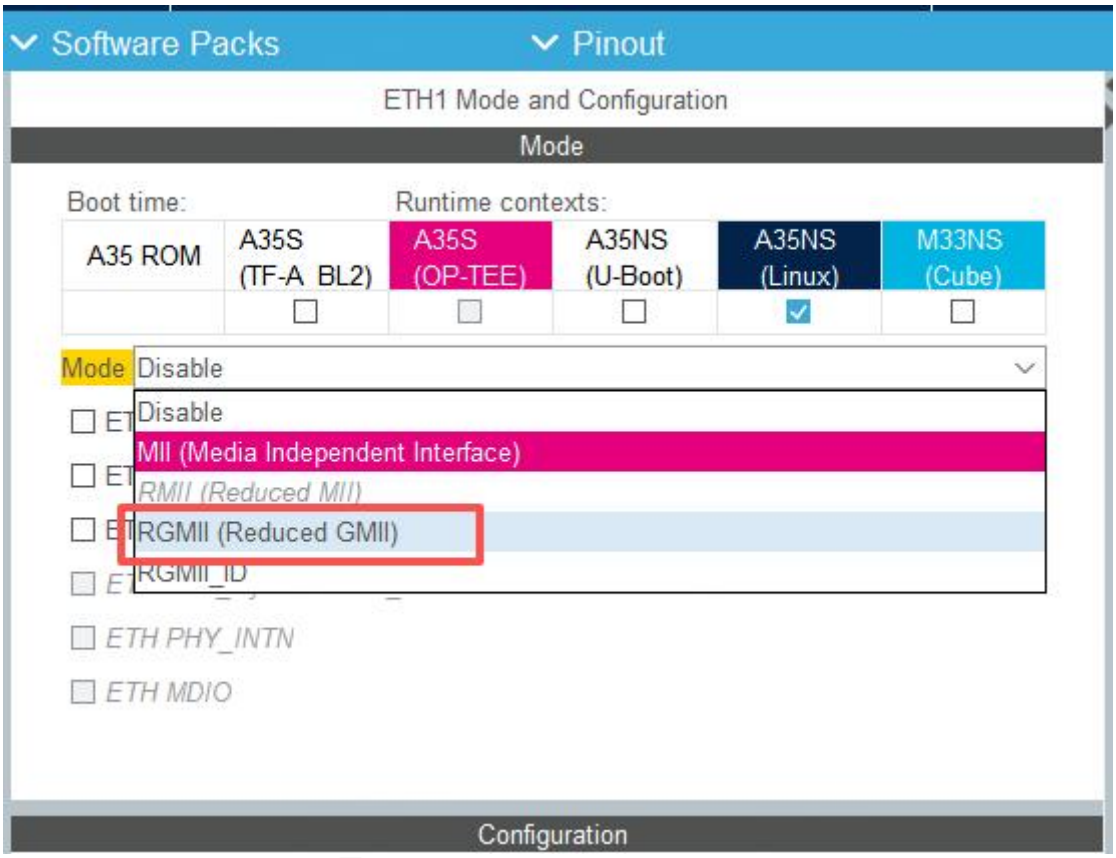
如上 cubemx 设置完成

4.2 以太网驱动开发

Cubemx6.14 打开 ic610.ioc，以太网使能及 gpio 复用设置在 Pinout&Configuration /Conntectivity 下。

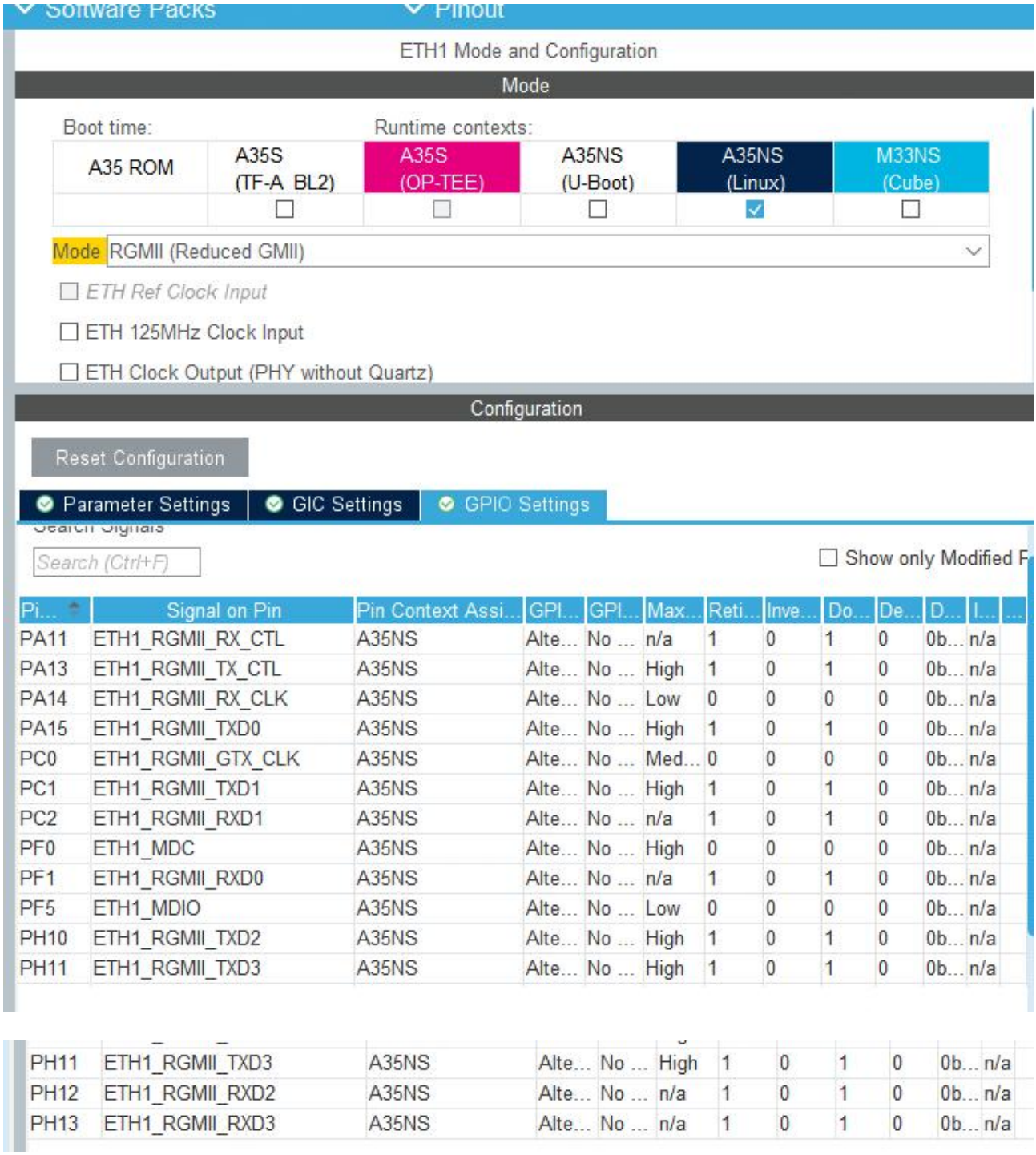


点击 ETH2，设置右侧 Mode，选择对应硬件模式，如当前硬件设计为千兆以太网，选择 RGMII



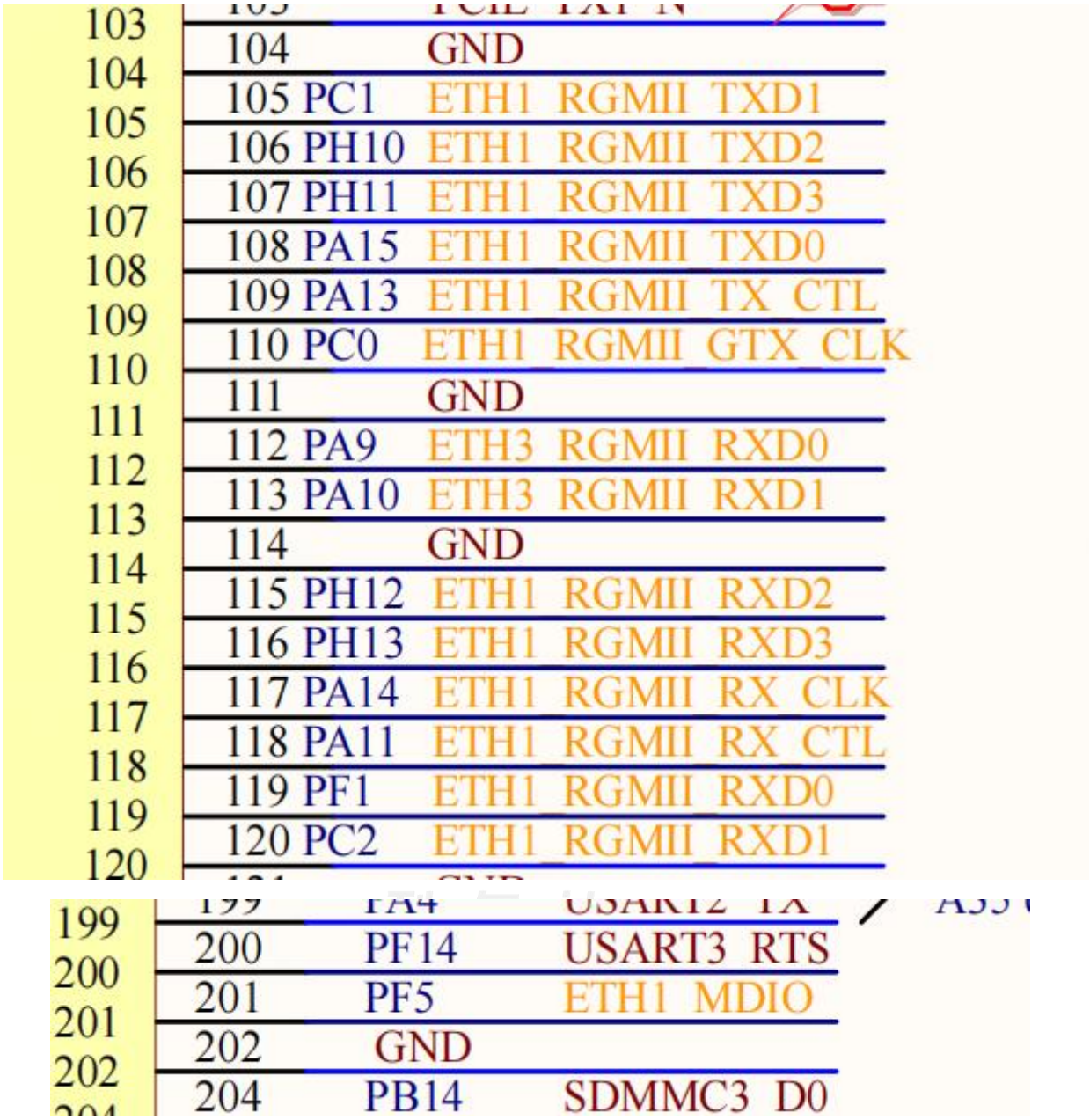
红色表示当前 gpio 存在冲突，该模式不能设置，可将鼠标放在红色区域根据提示信息排除冲突。

如上当前 cubemx 未使用的 gpio 可以复用为 RGMII，则该模式可以点击设置，设置后 cubemx 自动选择 gpio 并复用 gpio。



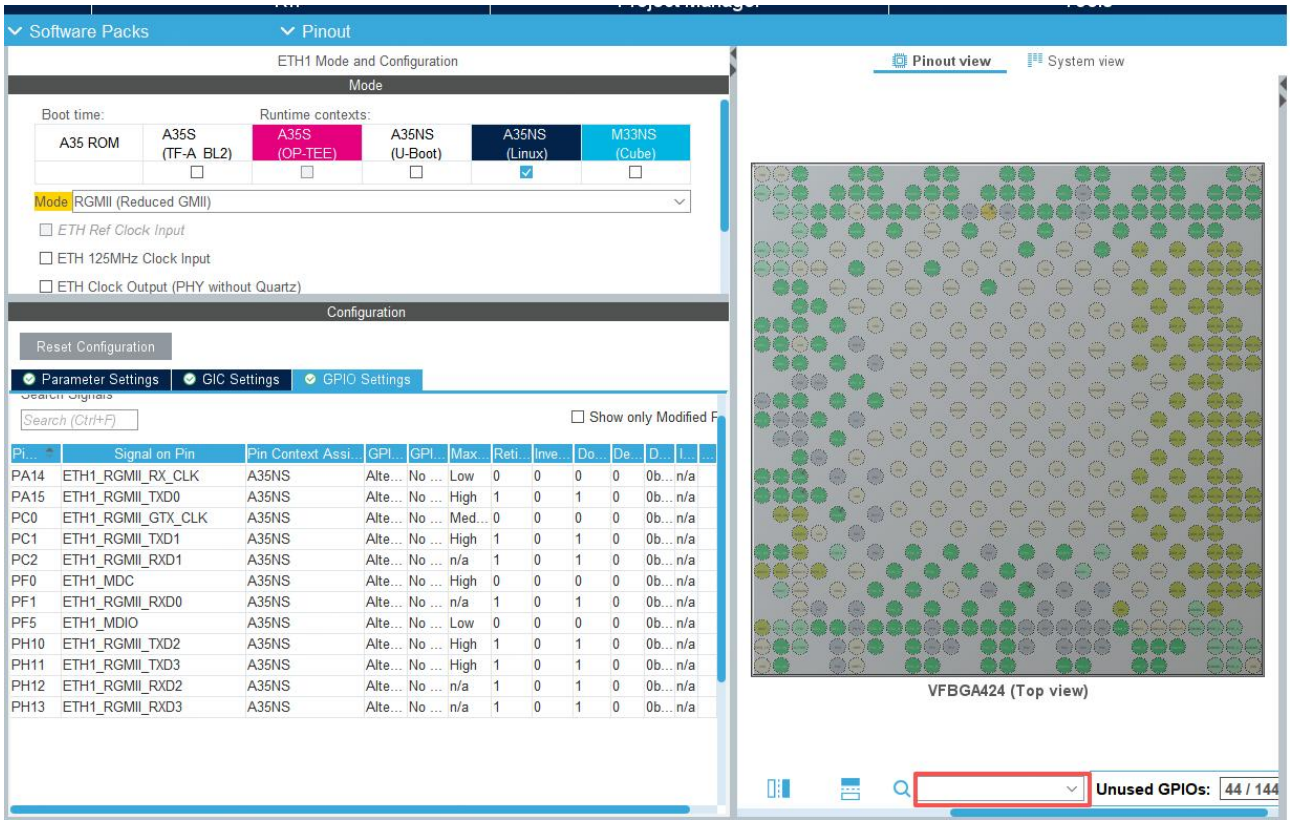
查看 GPIO Settings 下 gpio 复用情况，根据实际硬件设置就行修改。若有 Pin Name 与实际硬件设置不一致，可在 gpio 搜索框如下图，搜索 Pin Name 并设置复用功能。

硬件 eth1 gpio 复用情况：

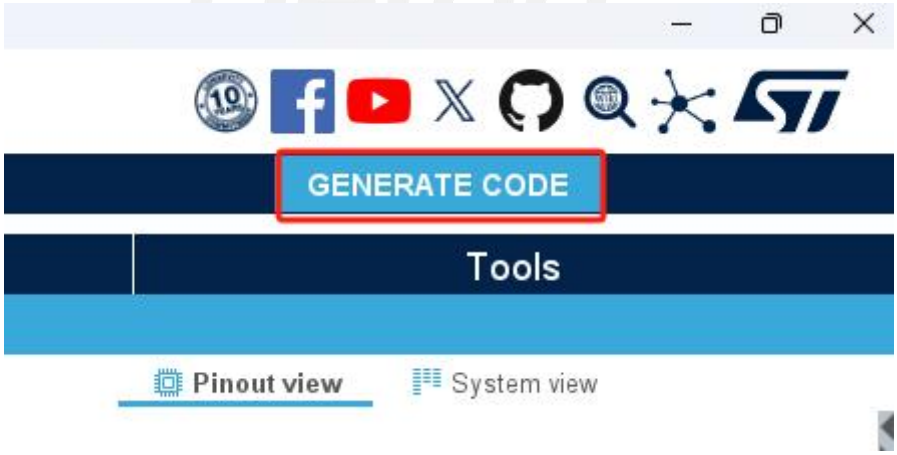


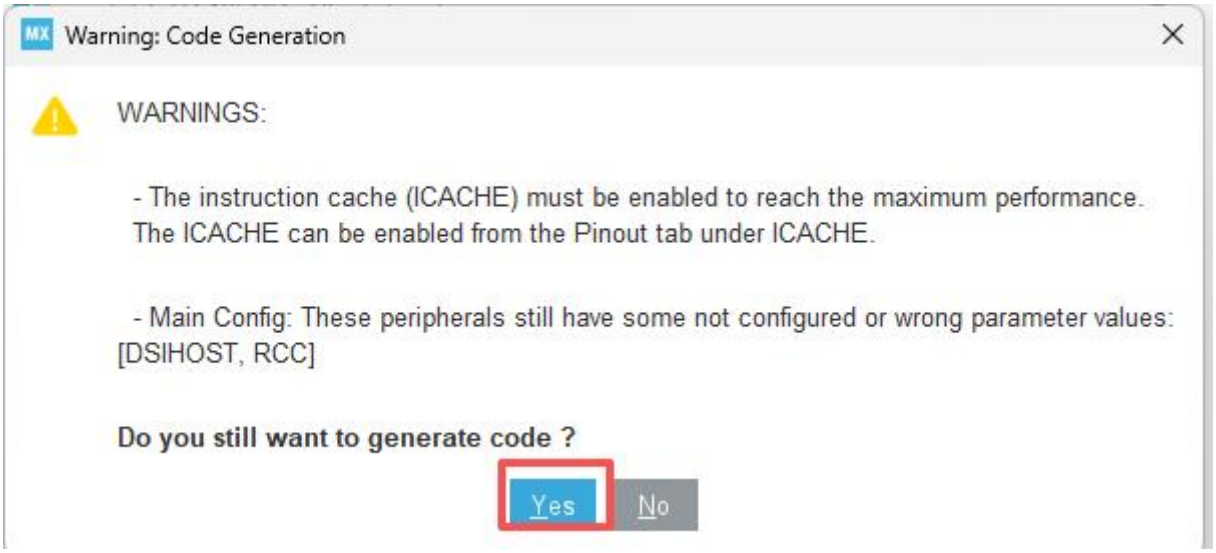
103			
104	104	GND	
105	105	PC1	ETH1_RGMII_TXD1
106	106	PH10	ETH1_RGMII_TXD2
107	107	PH11	ETH1_RGMII_TXD3
108	108	PA15	ETH1_RGMII_TXD0
109	109	PA13	ETH1_RGMII_TX_CTL
110	110	PC0	ETH1_RGMII_GTX_CLK
111	111	GND	
112	112	PA9	ETH3_RGMII_RXD0
113	113	PA10	ETH3_RGMII_RXD1
114	114	GND	
115	115	PH12	ETH1_RGMII_RXD2
116	116	PH13	ETH1_RGMII_RXD3
117	117	PA14	ETH1_RGMII_RX_CLK
118	118	PA11	ETH1_RGMII_RX_CTL
119	119	PF1	ETH1_RGMII_RXD0
120	120	PC2	ETH1_RGMII_RXD1

199			
200	200	PF14	USART3_RTS
201	201	PF5	ETH1_MDIO
202	202	GND	
204	204	PB14	SDMMC3_D0



Gpio 配置完成后点击 GENERATE CODE，即可更新 dts。





代码生成完成后，自动更新

ic610\CA35\DeviceTree\ic610\kernel\stm32mp255d-ic610-mx.dts

自动生成 gpio 复用及以太网 eth1 节点

```
    eth1_mdio_pins_mx: eth1_mdio_mx-0 {
        pins1 {
            pinmux = <STM32_PINMUX('F', 0, AF10)>; /* ETH1_MDC */
            bias-disable;
            drive-push-pull;
            slew-rate = <2>;
            st,io-clk-edge = <0>;
            st,io-retime = <0>;
            st,io-delay-path = <0>;
            st,io-delay = <0>;
        };
        pins2 {
            pinmux = <STM32_PINMUX('F', 5, AF8)>; /* ETH1_MDIO */
            bias-disable;
            drive-push-pull;
            slew-rate = <0>;
            st,io-clk-edge = <0>;
            st,io-retime = <0>;
            st,io-delay-path = <0>;
            st,io-delay = <0>;
        };
    };

    eth1_mdio_sleep_pins_mx: eth1_mdio_sleep_mx-0 {
        pins {
            pinmux = <STM32_PINMUX('F', 0, ANALOG)>, /* ETH1_MDC */
                     <STM32_PINMUX('F', 5, ANALOG)>; /* ETH1_MDIO */
        };
    };
```

```
&eth1 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&eth1_mdio_pins_mx>, <&eth1_rgmii_pins_mx>;
    pinctrl-1 = <&eth1_mdio_sleep_pins_mx>, <&eth1_rgmii_sleep_pins_mx>;
    status = "okay";

    /* USER CODE BEGIN eth1 */
    /* USER CODE END eth1 */
};
```

在 aliases 下添加 eth1 如下

```
aliases {

  ethernet1 = &eth1;

 };
```

在以下区域内添加 phy 相关配置信息即可。

USER CODE 区域为用户手动添加代码区域，该区域内不会被 cubemx 修改。

```
&eth1 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&eth1_mdio_pins_mx>, <&eth1_rgmii_pins_mx>;
    pinctrl-1 = <&eth1_mdio_sleep_pins_mx>, <&eth1_rgmii_sleep_pins_mx>;
    status = "okay";

    /* USER CODE BEGIN eth1 */
    /* USER CODE END eth1 */
};
```

```
/* USER CODE BEGIN eth1 */

phy-mode = "rgmii-id";

 max-speed = <1000>;

 phy-handle = <&phy1_eth1>;

 st,eth-clk-sel; //125M from rcc

 snps,ext-systime;

 //st,eth-ref-clk-sel;


 mdio1 {

  #address-cells = <1>;

  #size-cells = <0>;

  compatible = "snps,dwmac-mdio";

  phy1_eth1: ethernet-phy@0 {

   //compatible = "ethernet-phy-id001c.c916";

   compatible = "ethernet-phy-id937c.4032"; //jl2121
```

```
reset-gpios =    <&gpiob 15 GPIO_ACTIVE_LOW>;

reset-assert-us = <1000>;

reset-deassert-us = <10000>;

realtek,eee-disable;

jl2xx1,xmii-enable = <JL2XX1_SELECT_RGMII_STATIC_OP_EN>;

                                jl2xx1,xmii-tx_delay = <0>;

                                jl2xx1,xmii-rx_delay = <1>;


jl2xx1,init-access_type = <JL2XX1_SELECT_ACCESS_TYPE_C22>;

jl2xx1,init-interface_mode = <JL2XX1_SELECT_INTERFACE_UTP_RGMII>;

jl2xx1,downshift-enable = <JL2XX1_SELECT_DSFT_STATIC_OP_EN>;

jl2xx1,downshift-count = <4>;

jl2xx1,speed-enable = <(JL2XX1_SELECT_SPEED_STATIC_OP_EN | \

    JL2XX1_SELECT_SPEED_1000M)>;

jl2xx1,lpbk-enable = <JL2XX1_SELECT_LPBK_STATIC_OP_EN>;

jl2xx1,lpbk-mode = <JL2XX1_SELECT_LPBK_PCS>;

jl2xx1,clk-enable = <(JL2XX1_SELECT_CLK_STATIC_OP_EN | \

      JL2XX1_SELECT_CLK0_OP_EN | \

      JL2XX1_SELECT_CLK0_OUT_125M_EN)>;

jl2xx1,intr-enable = <(JL2XX1_SELECT_INTR_STATIC_OP_EN | \
```

```
            JL2XX1_SELECT_INTR_LINK_CHANGE_EN | \
            JL2XX1_SELECT_INTR_AN_COMPLETE_EN | \
            JL2XX1_SELECT_INTR_PAGE_RECEIVED_EN | \
            JL2XX1_SELECT_INTR_AN_ERROR_EN)>;
    jl2xx1,wol-enable = <(JL2XX1_SELECT_WOL_STATIC_OP_EN | \
        JL2XX1_SELECT_WOL_ENABLE)>;
    jl2xx1,led-enable = <(JL2XX1_SELECT_LED_STATIC_OP_EN | \
            JL2XX1_SELECT_LED_FUNC_MODE_EN | \
            JL2XX1_SELECT_LED_POLARITY_EN | \
            JL2XX1_SELECT_LED_BLINK_RATE_EN)>;
    jl2xx1,led-func_mode = <(JL2XX1_SELECT_LED0_LINK10 | \
            JL2XX1_SELECT_LED0_ACTIVITY | \
            JL2XX1_SELECT_LED1_LINK100 | \
            JL2XX1_SELECT_LED1_ACTIVITY | \
            JL2XX1_SELECT_LED2_LINK1000 | \
            JL2XX1_SELECT_LED2_ACTIVITY)>;
    jl2xx1,led-polarity = <(JL2XX1_SELECT_LED0_POLA_LOW | \
        JL2XX1_SELECT_LED1_POLA_LOW | \
        JL2XX1_SELECT_LED2_POLA_LOW)>;
    jl2xx1,led-blinkrate = <JL2XX1_SELECT_LED_BLINK_RATE_MS>;
```

```
    reg = <0>;
  };
 };/* USER CODE END eth1 */
```

Usercode 参数说明：

千兆以太网

```
phy-mode = "rgmii-id";
max-speed = <1000>;
```

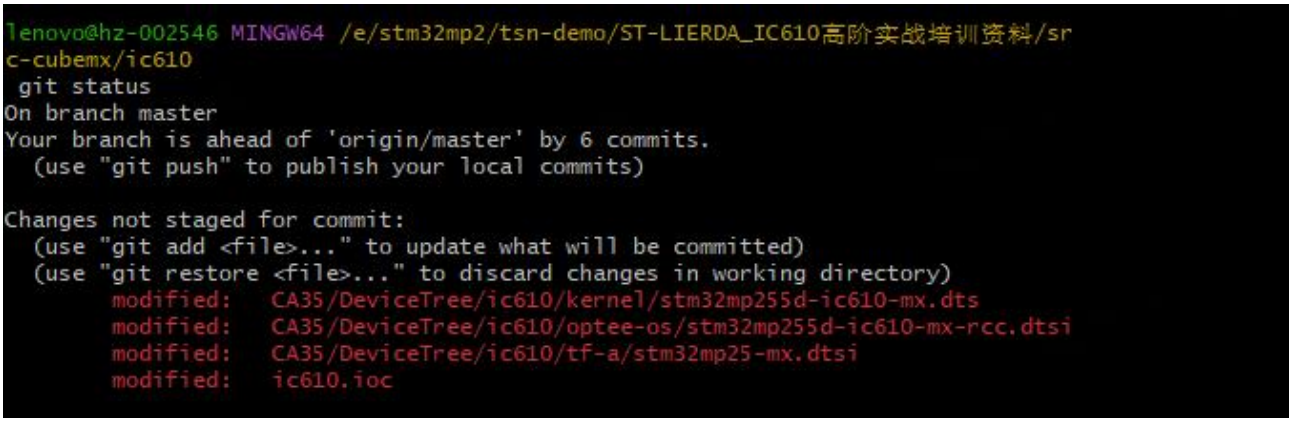
百兆以太网

```
phy-mode = "rmii";
max-speed = <100>;
```

st,eth-clk-sel; //使用 rcc 的 125M 代替 ETH_CLK125，必须使能该参数。

st,eth-ref-clk-sel; //使用 RCC clock instead of ETH_REF_CLK，必须使能该参数。

配置完成后可通过 git 查询变更的文件



其中，ic610.ioc 文件为配置文件，tf-a 下 stm32mp25-mx.dtsi 为 ddr 配置文件，每次生成会更新时间信息，无需关注。

Optee 和 kernel 下文件为需要更新文件

将最终的 ic610\CA35\DeviceTree\ic610\kernel\stm32mp255d-ic610-mx.dts 和 CA35/DeviceTree/ic610/optee-os/stm32mp255d-ic610-mx-rcc.dtsi 复制到 E:\vmware\下

在 Ubuntu2404 端：

Optee 下

```
cd optee-os-stm32mp-4.0.0-stm32mp-r1-r0/optee-os-stm32mp-4.0.0-stm32mp-r1/
optee-os-stm32mp-4.0.0-stm32mp-r1$ cp /mnt/hgfs/vmware/stm32mp255d-ic610-mx-rcc.dtsi ./core/arch/arm/dts/
optee-os-stm32mp-4.0.0-stm32mp-r1$ ./build.sh
```

编译完成后镜像为

../../ic610-images/fip/fip-stm32mp255d-ic610-mx-optee-emmc.bin

```
optee-os-stm32mp-4.0.0-stm32mp-r1$ cp ../../ic610-images/fip/fip-stm32mp255d-ic610-mx-optee-emmc.bin /mnt/hgfs/vmware/
```

Kernel 下：

```
cd ~/ic610/src-ic610/linux-6.6.48/linux-6.6.48
linux-6.6.48$ cp /mnt/hgfs/vmware/stm32mp255d-ic610-mx.dts arch/arm64/boot/dts/st/
```

替换原文件 stm32mp255d-ic610-mx.dts

```
linux-6.6.48$: ./build.sh
```

编译完成后镜像为

```
linux-6.6.48$: ../../ic610-images/bootfs/
```

Image.gz stm32mp255d-ic610-mx.dtb stm32mp257f-dk.dtb stm32mp257f-ev1.dtb

镜像自动更新到烧录镜像 st-image-bootfs-openstlinux-weston-stm32mp2.ext4

```
linux-6.6.48$ cp ../../ic610-images/st-image-bootfs-openstlinux-weston-stm32mp2.bootfs.ext4 /mnt/hgfs/vmware/
```

将 st-image-bootfs-openstlinux-weston-stm32mp2.ext4 和 fip-stm32mp255d-ic610-mx-optee-emmc.bin

复制到共享文件夹替换烧录镜像包 sdk_v24.11.06-stm32mp255d-evk-v2\下原文件，再次进入烧录模式 tsv 文件选择 update_eth1_images.tsv，不烧录文件系统，提高烧录速度。

4.3 以太网用户层操作

板子启动后

```
ifconfig -a
```

可查看全部网络设备节点，如以太网、can、4g 模组网卡等

```
root@stm32mp2:~# ifconfig -a
can0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:62

can1      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:63

can2      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:64

end0      Link encap:Ethernet  HWaddr 12:35:5A:BE:72:FE
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:67

end1      Link encap:Ethernet  HWaddr 06:70:1C:E8:FB:3B
          inet addr:192.168.2.144  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:655 errors:0 dropped:0 overruns:0 frame:0
          TX packets:253 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:55511 (54.2 KiB)  TX bytes:54773 (53.4 KiB)
          Interrupt:65 Base address:0x8000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:52 errors:0 dropped:0 overruns:0 frame:0
          TX packets:52 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3539 (3.4 KiB)  TX bytes:3539 (3.4 KiB)
```

默认镜像已经关闭 end1，若 eth1 相关配置更新成功，则会出现 end1 节点。

在 ic610 下以太网网卡节点为 endN。

如设置 end1 网卡静态 ip

```
root@stm32mp2:~# ifconfig end1 192.168.1.10

root@stm32mp2:~# ifconfig end1 up
```

```
root@stm32mp2:~# ifconfig end1
```

如下表示静态 ip 地址设置成功

```
root@stm32mp2:~#
root@stm32mp2:~#  ifconfig end1 192.168.1.10
root@stm32mp2:~# ifconfig end1 up
root@stm32mp2:~#  ifconfig end1
end1      Link encap:Ethernet  HWaddr C6:17:E2:72:F6:3F
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::c417:e2ff:fe72:f63f/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:86 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8480 (8.2 KiB)  TX bytes:7079 (6.9 KiB)
          Interrupt:69
```

将板子的 end1 网口与 pc 通过网线连接，pc 端设置静态 ip 地址 192.169.1.1。

板子的 end1 网口和 pc 连接后板子端自动打印如下：

```
          Interrupt:69
root@stm32mp2:~# [  154.251452] stm32-dwmac 482c0000.eth1 end1: Link is Up - 1Gbps/Full - flow control rx/tx
[  155.274866] stm32-dwmac 482c0000.eth1 end1: Link is Down
[  156.303436] stm32-dwmac 482c0000.eth1 end1: Link is Up - 1Gbps/Full - flow control rx/tx
```

```
root@stm32mp2:~# ping -I end1 192.168.1.1
```

如下表示板子与 pc 通信正常。

```
root@stm32mp2:~#
root@stm32mp2:~#
root@stm32mp2:~# ping -I end1 192.168.1.1
PING 192.168.1.1 (192.168.1.1) from 192.168.1.10 end1: 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=128 time=0.746 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=128 time=1.10 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=128 time=0.667 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=128 time=0.586 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=128 time=0.670 ms
64 bytes from 192.168.1.1: icmp_seq=6 ttl=128 time=0.610 ms
64 bytes from 192.168.1.1: icmp_seq=7 ttl=128 time=0.763 ms
64 bytes from 192.168.1.1: icmp_seq=8 ttl=128 time=0.963 ms
64 bytes from 192.168.1.1: icmp_seq=9 ttl=128 time=0.968 ms
64 bytes from 192.168.1.1: icmp_seq=10 ttl=128 time=0.688 ms
```

网速测试：

Pc 端安装下载 iperf3.exe

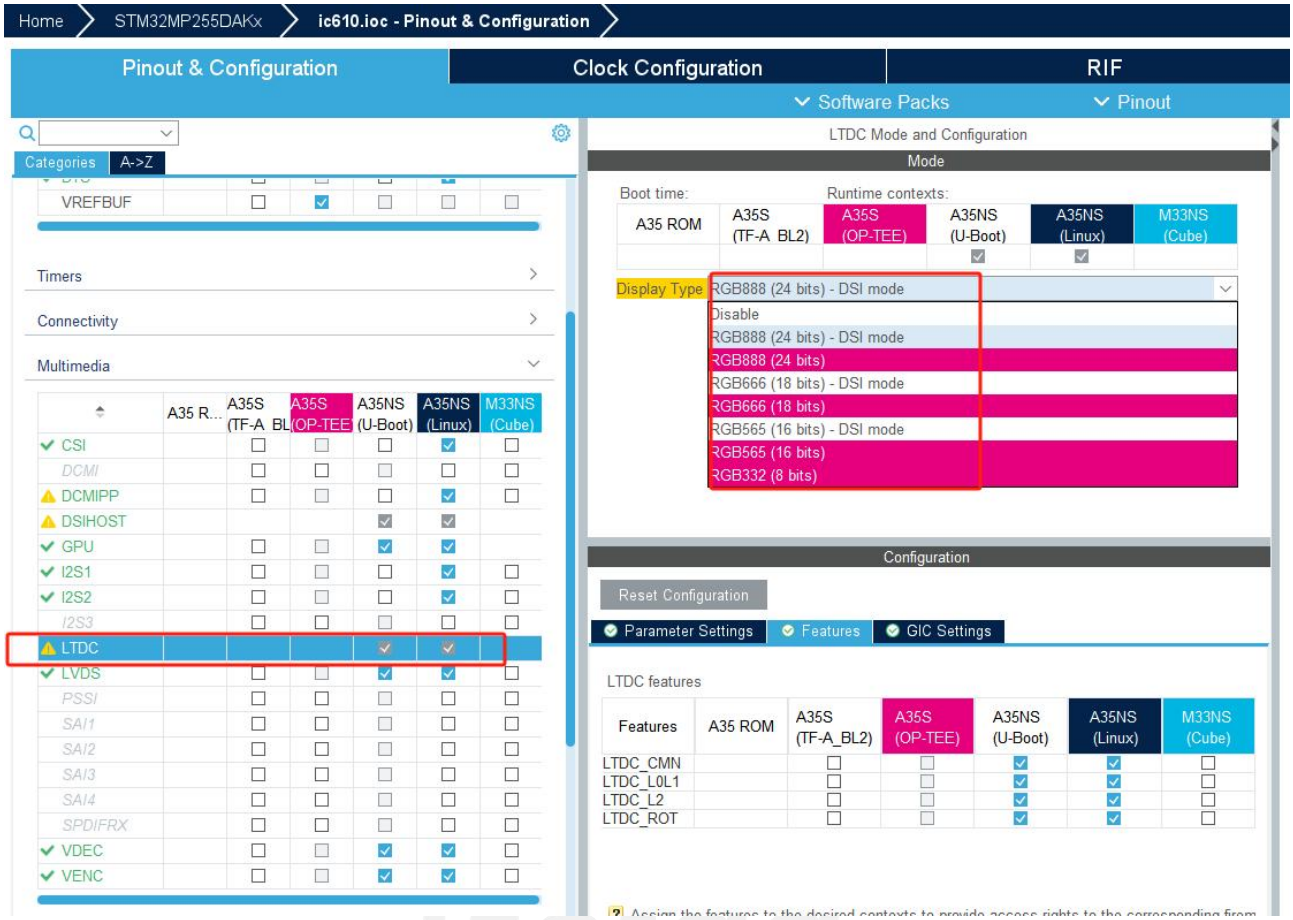
如 D:\iperf-3.1.3-win64\存在 iperf3.exe

5 IC610 DSI MIPI 显示模块的移植

5.1 IC610 显示驱动开发

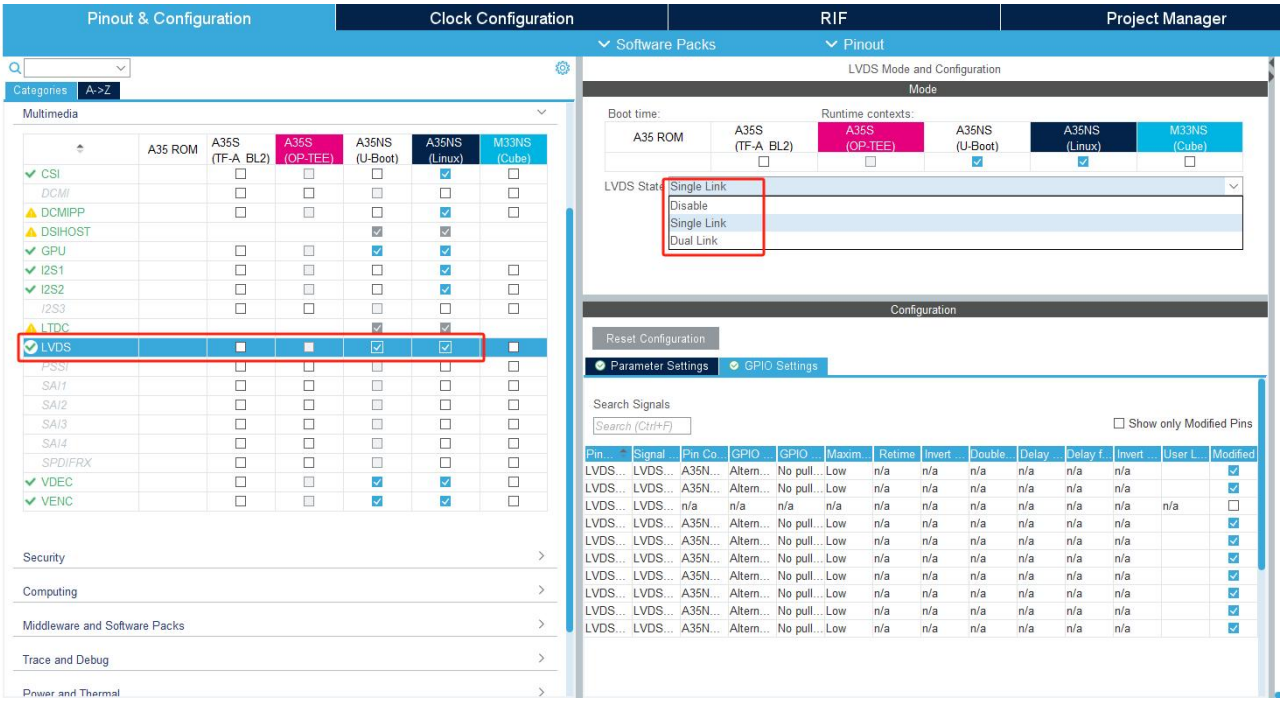
IC610 支持 rgb、lvds 和 dsi，且只支持单屏显示。

Cubemx6.14 打开 ic610.ioc，LTDC 设置界面如下。

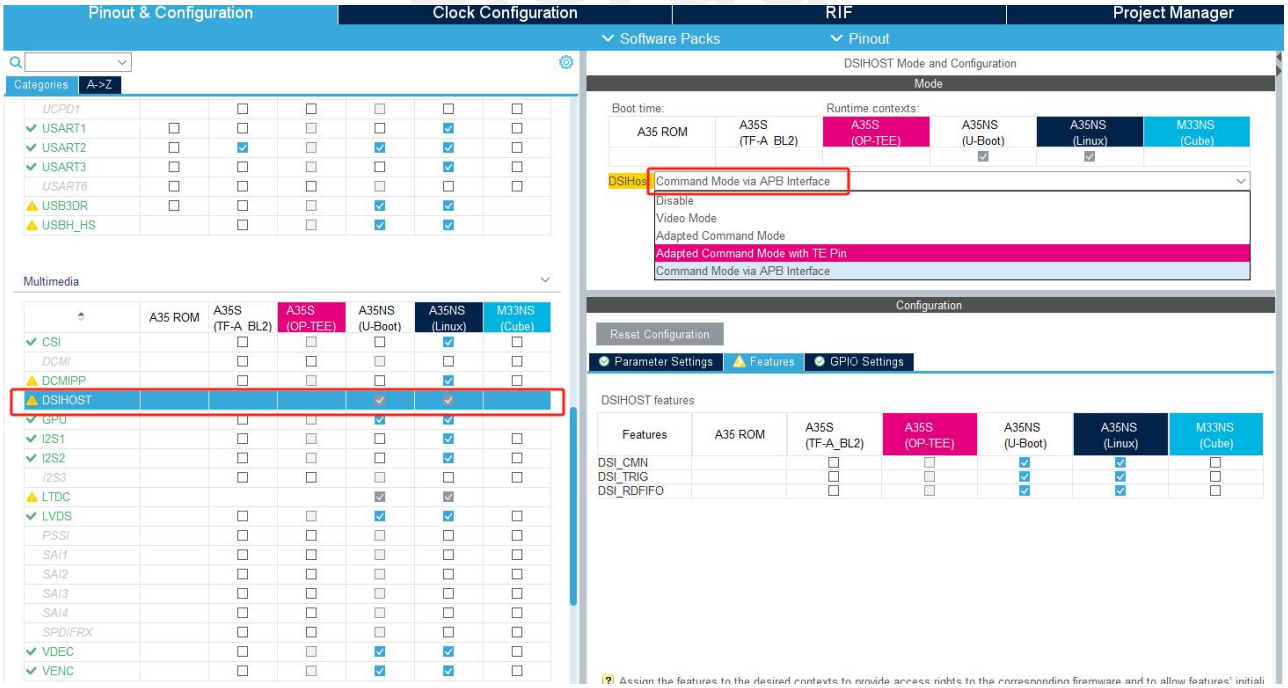


RGB：使用 ltdc 模块，Display Type 可支持 RGB888、RGB666、RGB565、RGB332，该模式下使用的是 gpio 复用功能，如 PF6 的复用功能 LTDC_B0。使用该模式下则 lvds、mipi 等无法使用且占用 gpio。DSI_mode

LVDS：CPU 自带 lvds 专用信号通道引脚，不占用 gpio，lvds 支持单通道、双通道屏幕，及单通道双屏同显。使用 lvds 时同样依赖 LTDC，此时 LTDC 需要设置在 dsi 模式下，当前 cubemx 下 RGB888-DSI mode、RGB666-DSI mode、RG565-DSI mode 无差异，任意配置其中一种即可。Lvds 在 cubemx 设置如下，支持 Single Link 和 Dual Link,当前 cubemx 下设置任意模式即可，Single Link 和 Dual Link 需要在 usercode 中设置。



Dsi：CPU 自带 DSI 信号通道引脚，支持 mipi 显示可外接 mipi 屏，同时可使用 mipi 转 hdmi 芯片实现 hdmi 显示。DSI 默认使用 Command Mode via APB Interface 即可使用 cpu 自带 dsi 信号引脚。



5.2 dts 设置

在 ic610 的 kernel stm32mp255d-ic610-mx.dts 下通过修改配置实现显示切换。

```
#define display_value 1
```

// 1:lvds 单通道 与 hdmi 2:mipi

Lvds dts 单通道 USER CODE 设置如下：

```
#ifdef en_lvds
 panel_lvds: panel-lvds {
  compatible = "edt,etml0700z9ndha", "panel-lvds";
  //enable-gpios = <&gpiog 15 GPIO_ACTIVE_HIGH>;
  backlight = <&panel_lvds_backlight>;
  status = "okay";

  width-mm = <217>;
  height-mm = <135>;
  data-mapping = "vesa-24";

  panel-timing {
   clock-frequency = <72400000>;
   hactive = <1280>;
   vactive = <800>;
```

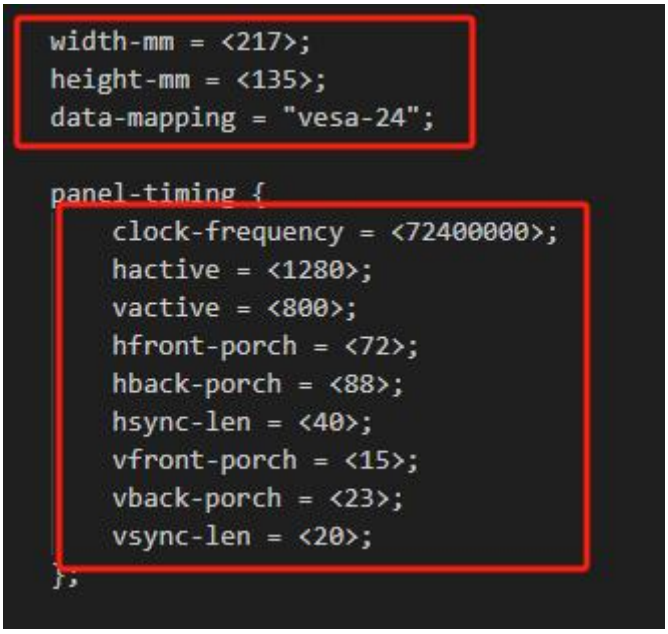
```
    hfront-porch = <72>;
    hback-porch = <88>;
    hsync-len = <40>;
    vfront-porch = <15>;
    vback-porch = <23>;
    vsync-len = <20>;
   };

   port {
    lvds_panel_in: endpoint {
     remote-endpoint = <&lvds_out0>;
    };
   };
  };
  #endif

  panel_lvds_backlight: panel-lvds-backlight {
   compatible = "gpio-backlight";
   // P Pb7
   gpios = <&gpiob 7 GPIO_ACTIVE_HIGH>;
```

```
    default-on;
    default-brightness-level = <1>;
    status = "okay";
  };
```

enable-gpios = <&gpiog 15 GPIO_ACTIVE_HIGH>；为 lvds 使能控制引脚，当前配套的 lvds 无此引脚故注释即可，该配置可根据实际硬件 gpio 进行设置。

panel_lvds_backlight 为 lvds gpio 背光控制驱动，根据时间硬件修改 gpio 即可。

和 lvds 屏幕相关参数主要有如下：



以上参数需要根据 lvds 屏幕数据手册进行设置，data-mapping 可根据数据手册实现与 linux 源码下文件 Documentation/devicetree/bindings/display/lvds.yaml 进行分析，判断 lvds 的 data-mapping。

Lvds dts 双通道 USER CODE 设置如下：

```
  panel_lvds: panel-lvds {
```

```
compatible = "edt,etml0700z9ndha", "panel-lvds";
//enable-gpios = <&gpiog 15 GPIO_ACTIVE_HIGH>;
backlight = <&panel_lvds_backlight>;
status = "okay";
#if 1 //G156HTN02 15.6 inch
width-mm = <364>;
height-mm = <216>;
data-mapping = "vesa-24";
  panel-timing {
 clock-frequency = <71100000>;
 hactive = <1920>;
 vactive = <1080>;
 hfront-porch = <75>;
 hback-porch = <75>;
 hsync-len = <10>;
 vfront-porch = <5>;
 vback-porch = <13>;
 vsync-len = <5>;
};
ports {
```

```
#address-cells = <1>;
#size-cells = <0>;

port@0 {
 reg = <0>;
  dual-lvds-odd-pixels;
 panel_lvds_in0: endpoint {
  remote-endpoint = <&lvds_out0>;
 };
};

port@1 {
 reg = <1>;
      dual-lvds-even-pixels;
   panel_lvds_in1: endpoint {
  remote-endpoint = <&lvds_out1>;
 };
};
};
};
```

```
&lvds {
 status = "okay";

 /* USER CODE BEGIN lvds */
 default-on;
 status = "disbaled";
 vdd-supply =    <&scmi_vddcore>;
 vdda18-supply = <&scmi_v1v8>;
  status = "okay";

 ports {
  #address-cells = <1>;
  #size-cells = <0>;

  port@0 {
   reg = <0>;
   lvds_in: endpoint {
    remote-endpoint = <&ltdc_ep0_out>;
```

```
    };
  };

  port@1 {
    reg = <1>;
    lvds_out0: endpoint {
      remote-endpoint = <&panel_lvds_in0>;
    };
  };

  port@2 {
    reg = <2>;
    lvds_out1: endpoint {
      remote-endpoint = <&panel_lvds_in1>;
    };
  };
 };
 /* USER CODE END lvds */
};
```

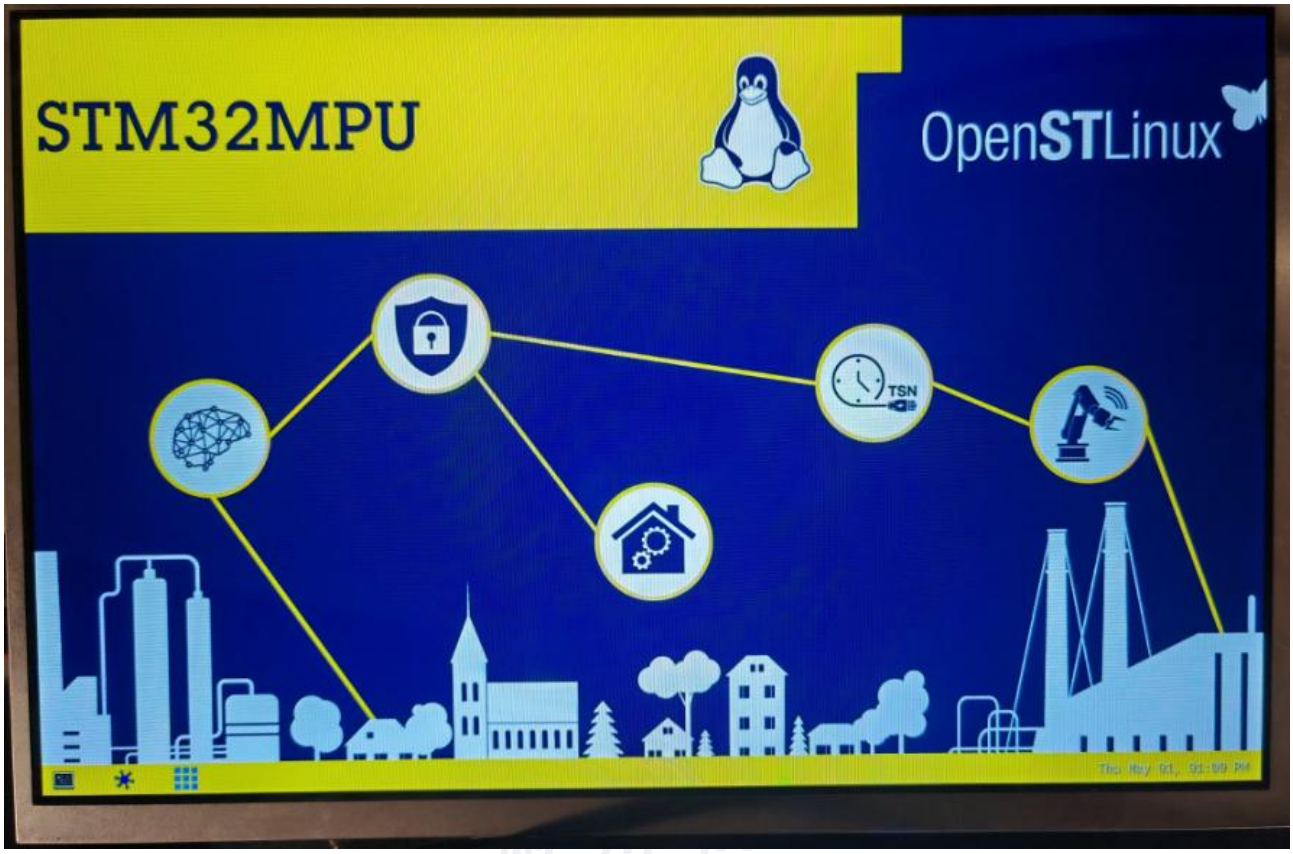
Dsi 屏幕根据选用的屏幕，修改 dts 中节点 panel_dsi 即可。

```
panel_dsi:panel-dsi@0 {
    compatible = "radxa,display-8hd-ad002";
            reg = <0>;
            //reset-gpios = <&gpioz 8 GPIO_ACTIVE_LOW>; //spi8 used
            //reset-gpios = <&gpioi 0 GPIO_ACTIVE_HIGH>; //real gpio display error
            vdd-supply = <&vdddo_3v3>;
            vccio-supply = <&vdddo_3v3>;

            backlight = <&panel_dsi_backlight>;
            status = "okay";

     port {
                    panel_in_dsi: endpoint {
                            remote-endpoint = <&dsi_out>;
                    };
            };

  };
```

```
  panel_dsi:panel-dsi@0 {
            compatible = "rocktech,hx8399";
        reg = <0>;
            reset-gpios = <&mcp23017 13 GPIO_ACTIVE_LOW>;
        power-supply = <&scmi_v3v3>;
        backlight = <&panel_dsi_backlight>;
        status = "okay";

        clock-frequency = <130854000>;
                hactive = <1080>;
                vactive = <1920>;
                hfront-porch = <35>;
                hback-porch = <10>;
                hsync-len = <5>;
                hsync-active = <0>;
                vfront-porch = <4>;
                vback-porch = <4>;
                vsync-len = <2>;
                vsync-active = <0>;

         port {
                        panel_in_dsi: endpoint {
                                remote-endpoint = <&dsi_out>;
                        };
                };
    };
```

5.3 屏幕测试

开机后文件系统默认显示 weston 桌面，如下图：



modetest -M stm 可查询 drm 相关节点信息。

```
root@stm32mp2:~# modetest -M stm
Encoders:
id      crtc    type    possible crtcs  possible clones
31      41      LVDS    0x00000001      0x00000001

Connectors:
id      encoder status          name            size (mm)       modes   encoders
32      31      connected       LVDS-1          217x135         1       31
  modes:
        index name refresh (Hz) hdisp hss hse htot vdisp vss vse vtot
  #0 1280x800 57.02 1280 1352 1392 1480 800 815 835 858 72400 flags: ; type: preferred, driver
  props:
        1 EDID:
                flags: immutable blob
                blobs:

                value:
        2 DPMS:
                flags: enum
                enums: On=0 Standby=1 Suspend=2 Off=3
                value: 0
        5 link-status:
                flags: enum
                enums: Good=0 Bad=1
                value: 0
        6 non-desktop:
                flags: immutable range
                values: 0 1
                value: 0
        4 TILE:
                flags: immutable blob
                blobs:

                value:
        33 dithering:
                flags: enum
                enums: off=0 on=1
                value: 0

CRTCs:
id      fb      pos     size
41      54      (0,0)   (1280x800)
  #0 1280x800 57.02 1280 1352 1392 1480 800 815 835 858 72400 flags: ; type: preferred, driver
  props:
        24 VRR_ENABLED:
                flags: range
                values: 0 1
                value: 0
        28 GAMMA_LUT:
                flags: blob
                blobs:

                value:
```

weston ui 关闭：

```
systemctl stop weston-graphical-session.service
```

重新打开 weston ui：

```
systemctl restart weston-graphical-session.service
```

使用 modetest 测试显示效果时需要先关闭 wenston。

如 lvds：

```
systemctl stop weston-graphical-session.service
```

```
modetest -M stm -s 32@41:1280x800
```

Mipi 下为：

```
systemctl stop weston-graphical-session.service

modetest -M stm -s 32@41:1080x1920
```

Hdmi 下：

```
systemctl stop weston-graphical-session.service

modetest -M stm -s 32:1920x1080-60 -v
```

6 IC610 启动及镜像烧录

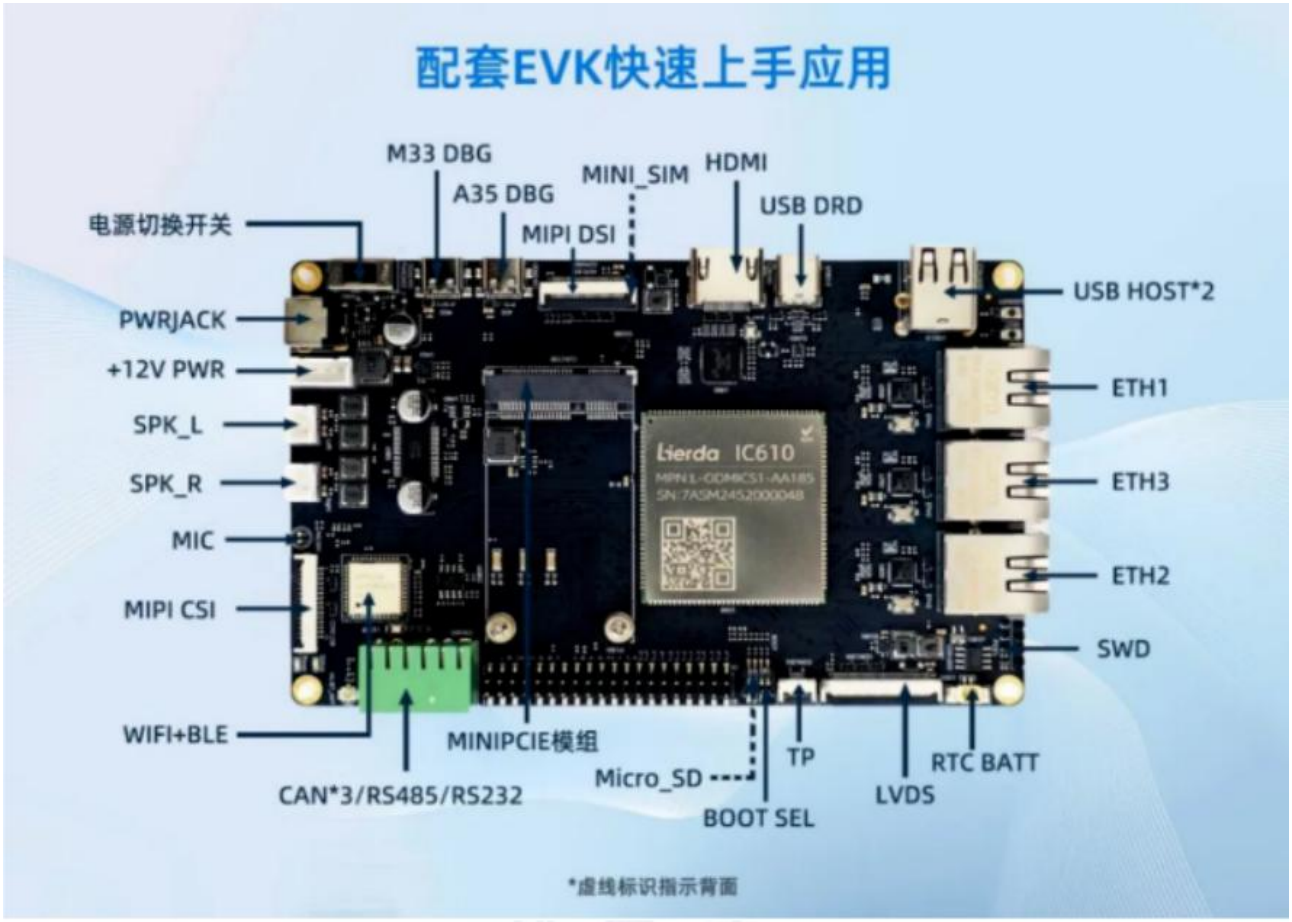


图 1.1 evk 正面实物图

如上为 evk 实物图，PWRJACK 为电源接口，接入 12V2A，拨动电源切换开关

6.1 STM32CubeProgrammer 安装

Windows 下双击 ST-LIERDA_IC610 高阶实战培训资料\软件\SetupSTM32CubeProgrammer_win64.exe 安装


STM32CubeProgrammer Installation Wizard
Welcome to the Installation of STM32CubeProgrammer 2.13.0
Step 1 of 9
STM32 CubeProgrammer
Starting STM32CubeProgrammer 2.13.0 installation
The homepage is at: http://www.st.com/en/development-tools/stm32cubeprog.html
life.augmented
STM32 Cube
STMicroelectronics
Next
Quit

图 2.1 烧录软件安装

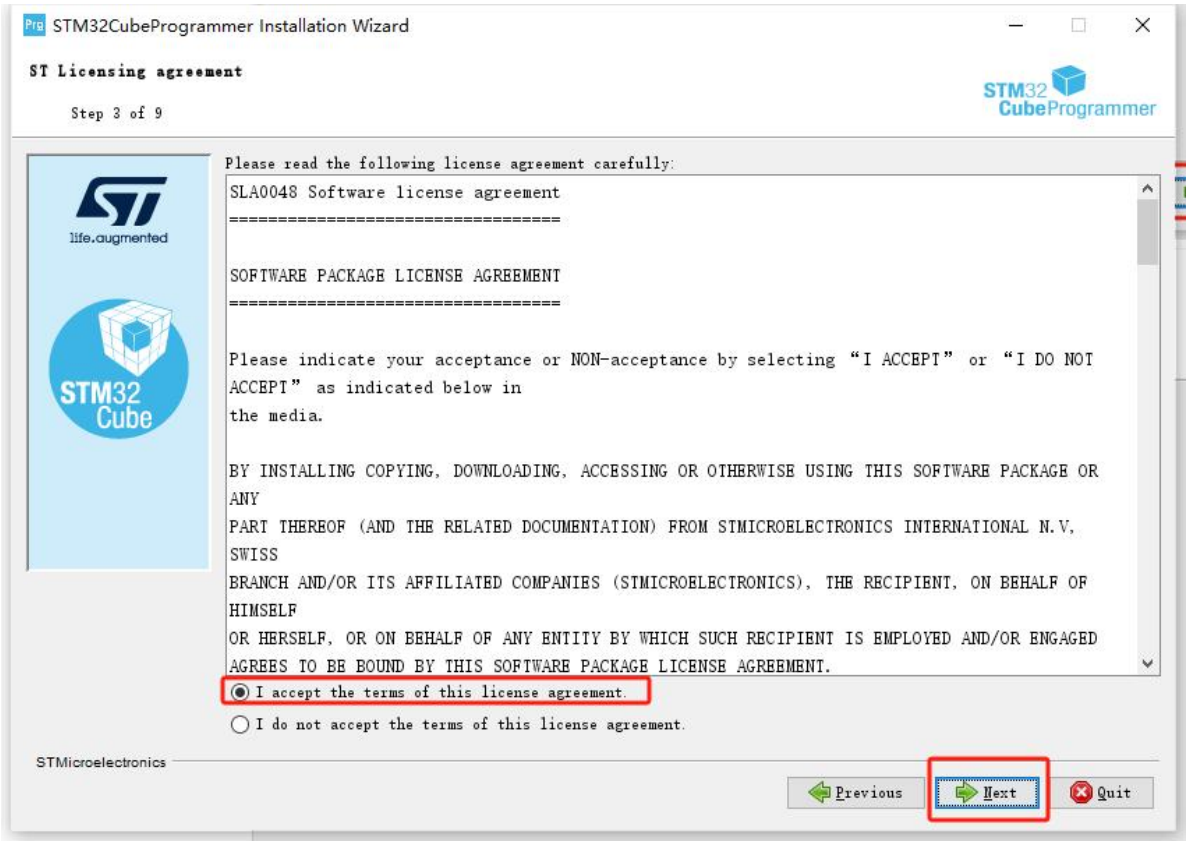

STM32CubeProgrammer Installation Wizard
ST Licensing agreement
Step 3 of 9
STM32 CubeProgrammer
life.augmented
STM32 Cube
Please read the following license agreement carefully:
SLA0048 Software license agreement
SOFTWARE PACKAGE LICENSE AGREEMENT
Please indicate your acceptance or NON-acceptance by selecting “I ACCEPT” or “I DO NOT ACCEPT” as indicated below in the media.
BY INSTALLING COPYING, DOWNLOADING, ACCESSING OR OTHERWISE USING THIS SOFTWARE PACKAGE OR ANY PART THEREOF (AND THE RELATED DOCUMENTATION) FROM STMICROELECTRONICS INTERNATIONAL N.V, SWISS BRANCH AND/OR ITS AFFILIATED COMPANIES (STMICROELECTRONICS), THE RECIPIENT, ON BEHALF OF HIMSELF OR HERSELF, OR ON BEHALF OF ANY ENTITY BY WHICH SUCH RECIPIENT IS EMPLOYED AND/OR ENGAGED AGREES TO BE BOUND BY THIS SOFTWARE PACKAGE LICENSE AGREEMENT.
I accept the terms of this license agreement.
I do not accept the terms of this license agreement.
STMicroelectronics
Previous
Next
Quit

图 2.2 接受协议


STM32CubeProgrammer Installation Wizard
Welcome to the Installation of STM32CubeProgrammer 2.13.0
Step 1 of 9
STM32 CubeProgrammer
life.augmented
STM32 Cube
Starting STM32CubeProgrammer 2.13.0 installation
The homepage is at: http://www.st.com/en/development-tools/stm32cubeprog.html
STMicroelectronics
Next
Quit

图 2.3 烧录软件安装

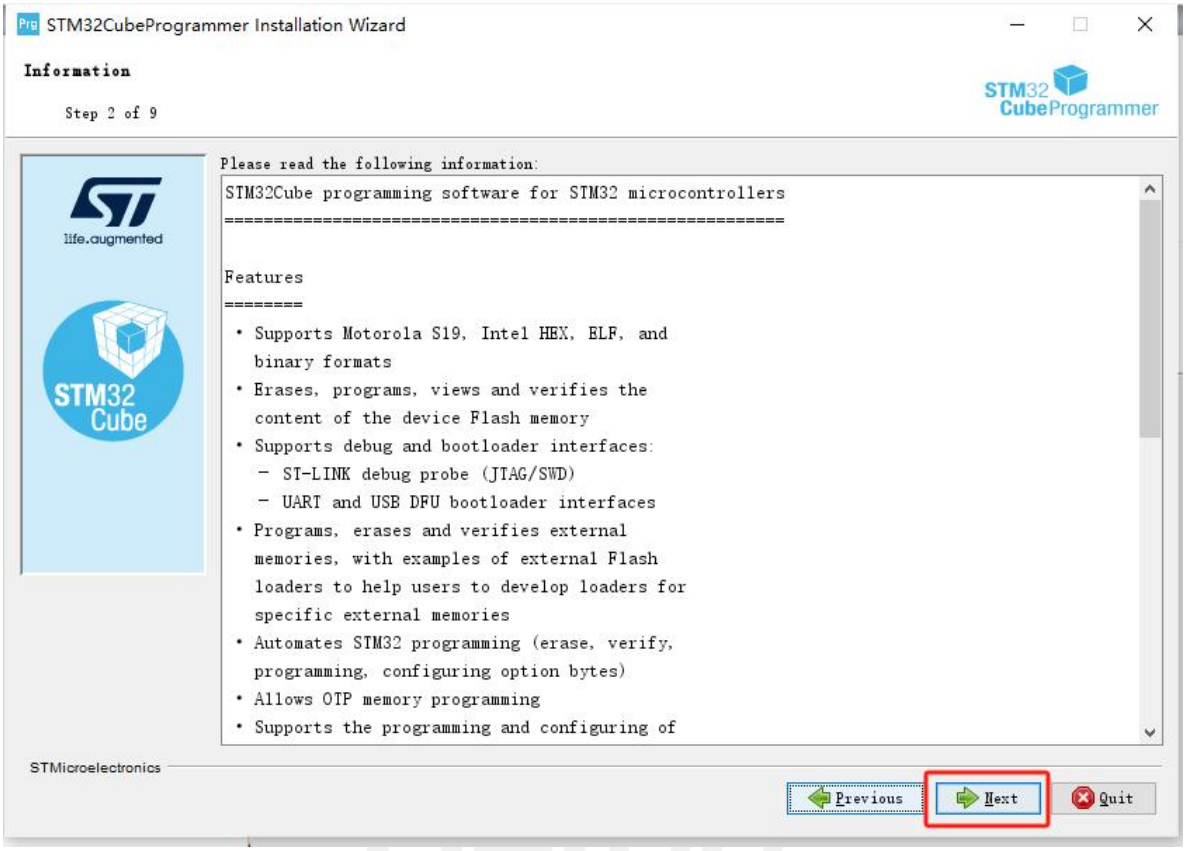


图 2.4 烧录软件安装

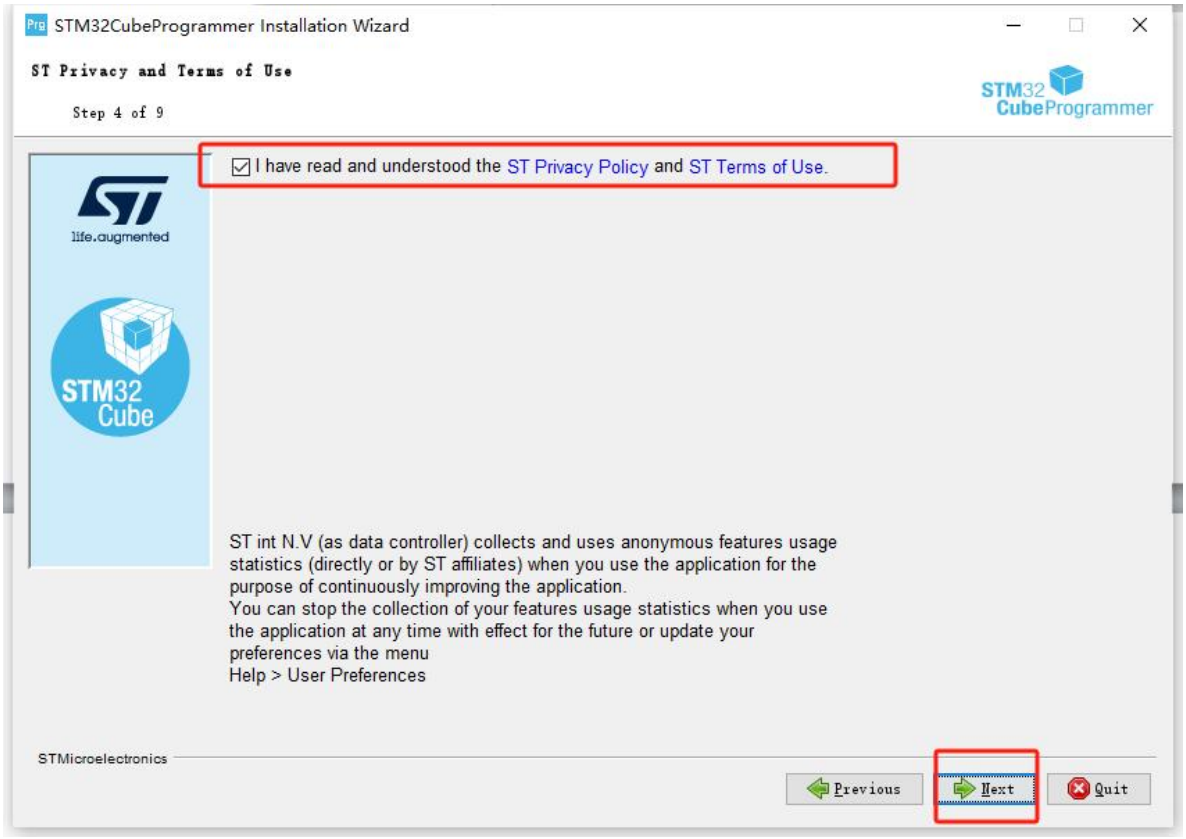

STM32CubeProgrammer Installation Wizard
ST Privacy and Terms of Use
Step 4 of 9
STM32 CubeProgrammer
I have read and understood the ST Privacy Policy and ST Terms of Use.
life.augmented
STM32 Cube
ST int N.V (as data controller) collects and uses anonymous features usage statistics (directly or by ST affiliates) when you use the application for the purpose of continuously improving the application.
You can stop the collection of your features usage statistics when you use the application at any time with effect for the future or update your preferences via the menu
Help > User Preferences
STMicroelectronics
Previous
Next
Quit

图 2.5 烧录软件安装

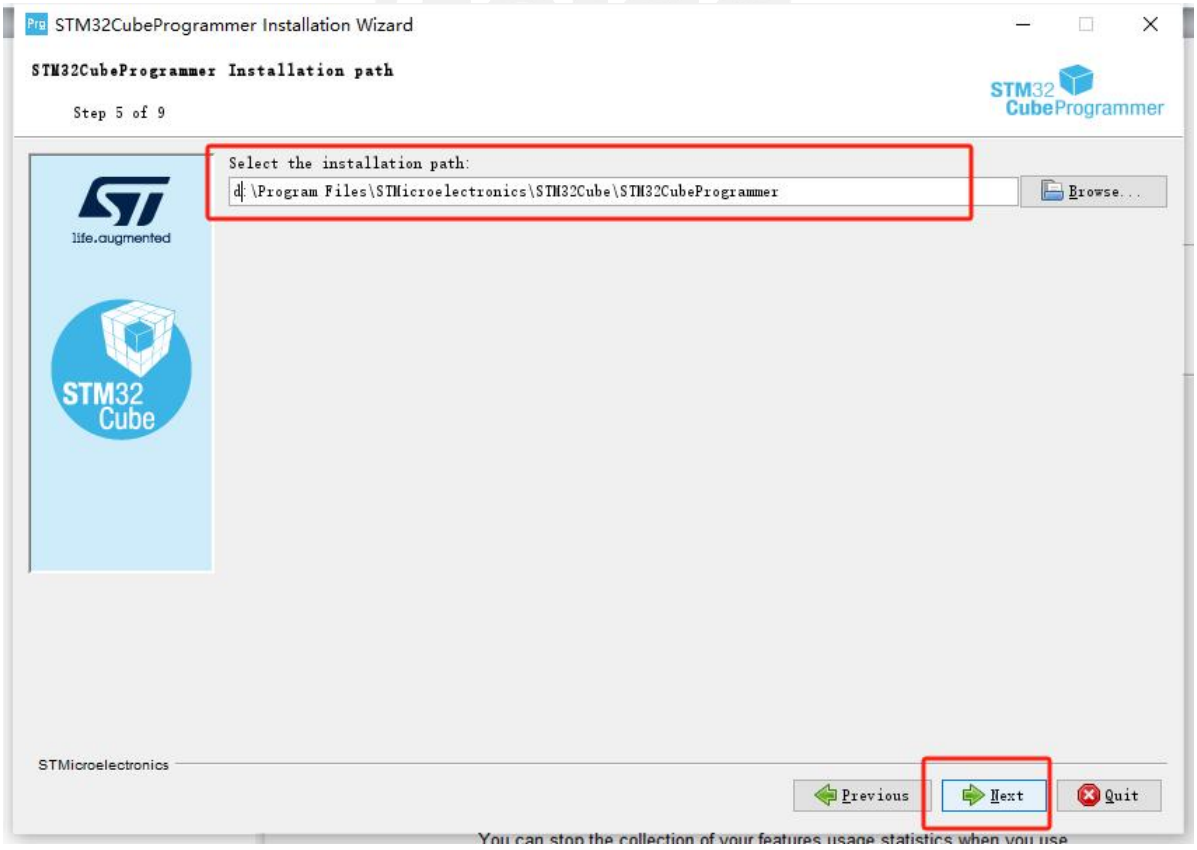

STM32CubeProgrammer Installation Wizard
STM32CubeProgrammer Installation path
Step 5 of 9
STM32 CubeProgrammer
Select the installation path:
d:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer
Browse...
life.augmented
STM32 Cube
STMicroelectronics
Previous
Next
Quit

图 2.6 烧录软件安装

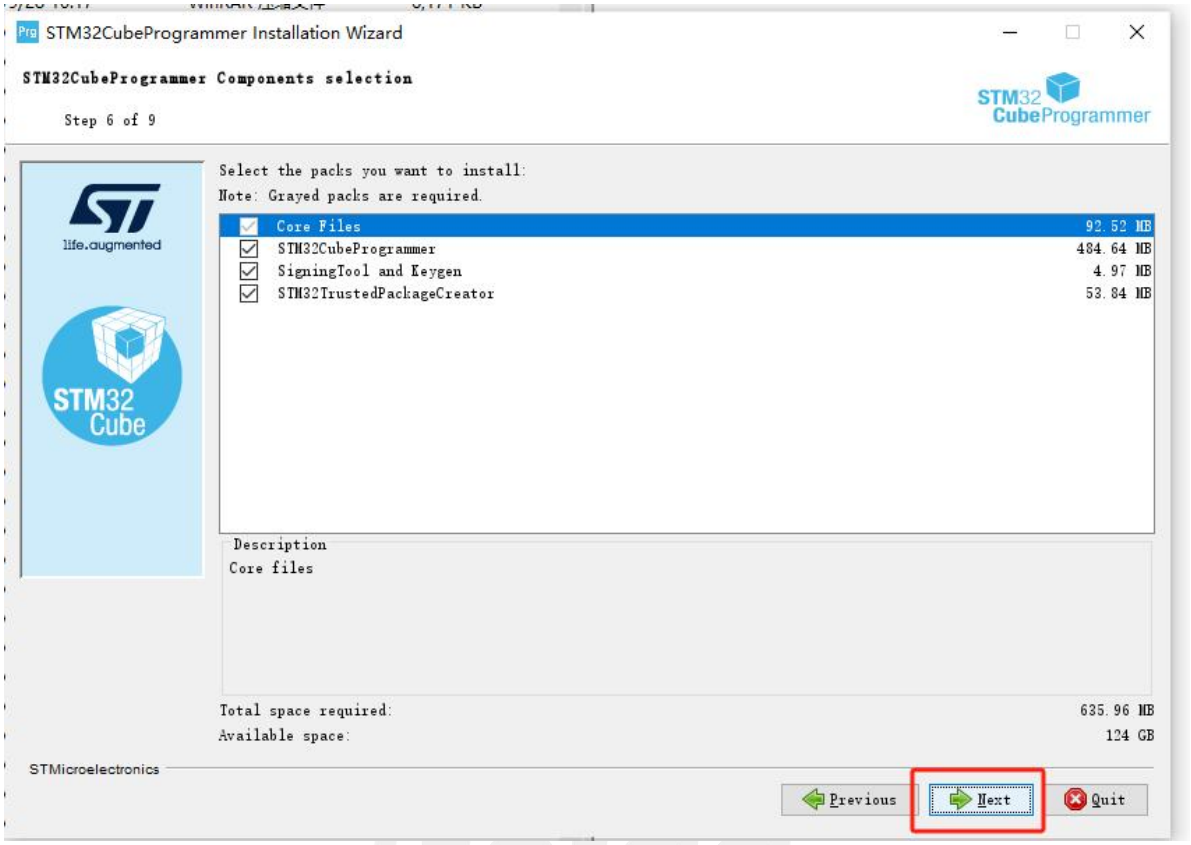


图 2.7 烧录软件安装

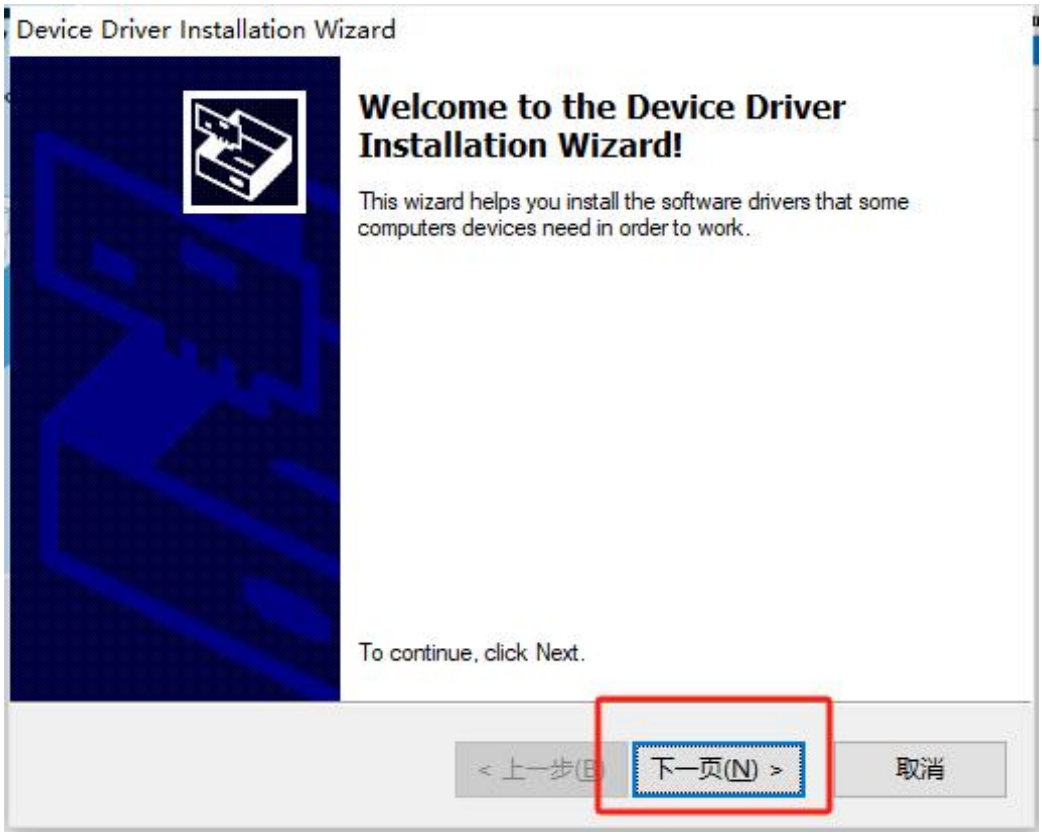

Device Driver Installation Wizard
Welcome to the Device Driver Installation Wizard!
This wizard helps you install the software drivers that some computers devices need in order to work.
To continue, click Next.
< 上一步(B)
下一页(N) >
取消

图 2.8 驱动安装

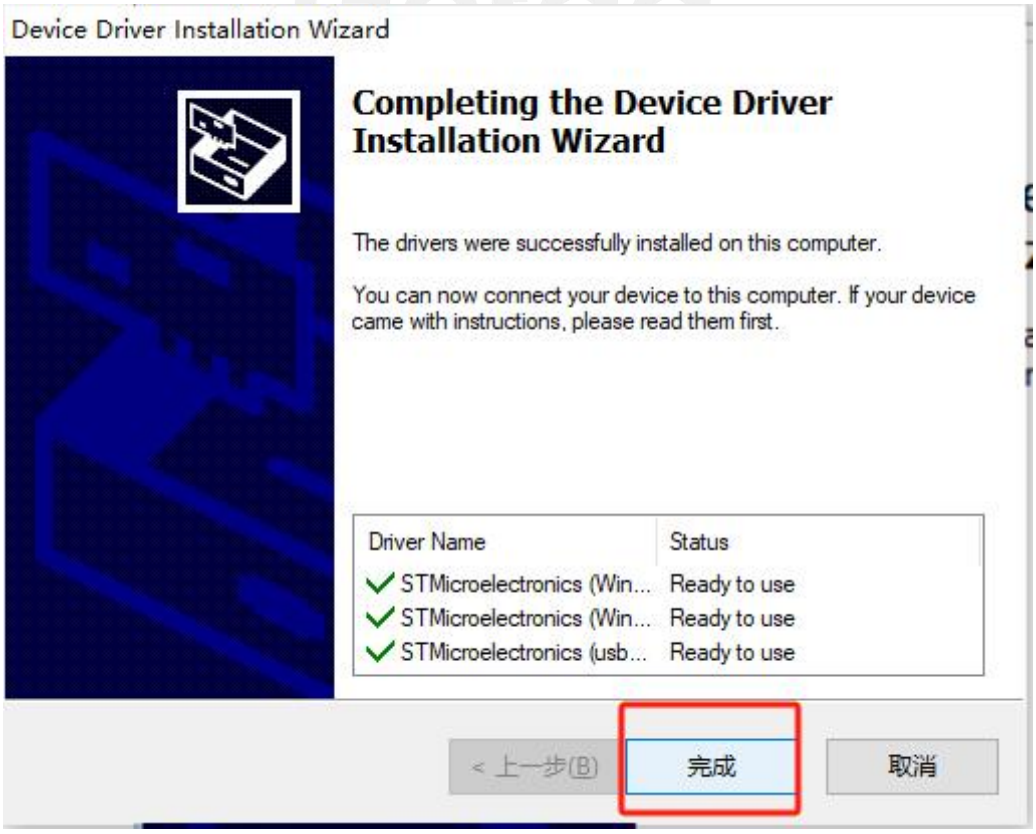

Device Driver Installation Wizard
Completing the Device Driver Installation Wizard
The drivers were successfully installed on this computer.
You can now connect your device to this computer. If your device came with instructions, please read them first.
Driver Name
Status
STMicroelectronics (Win...
Ready to use
STMicroelectronics (Win...
Ready to use
STMicroelectronics (usb...
Ready to use
< 上一步(B)
完成
取消

图 2.9 驱动安装完成

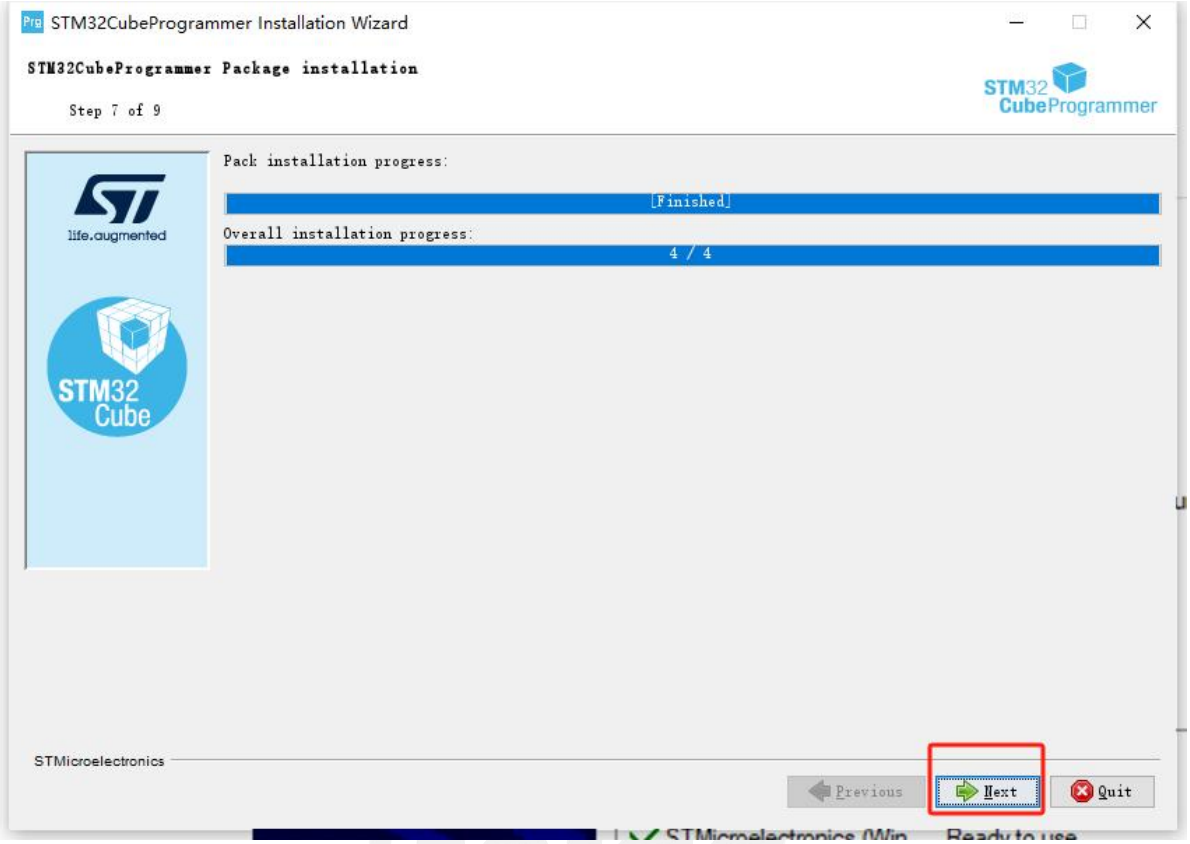


图 2.10 烧录软件安装

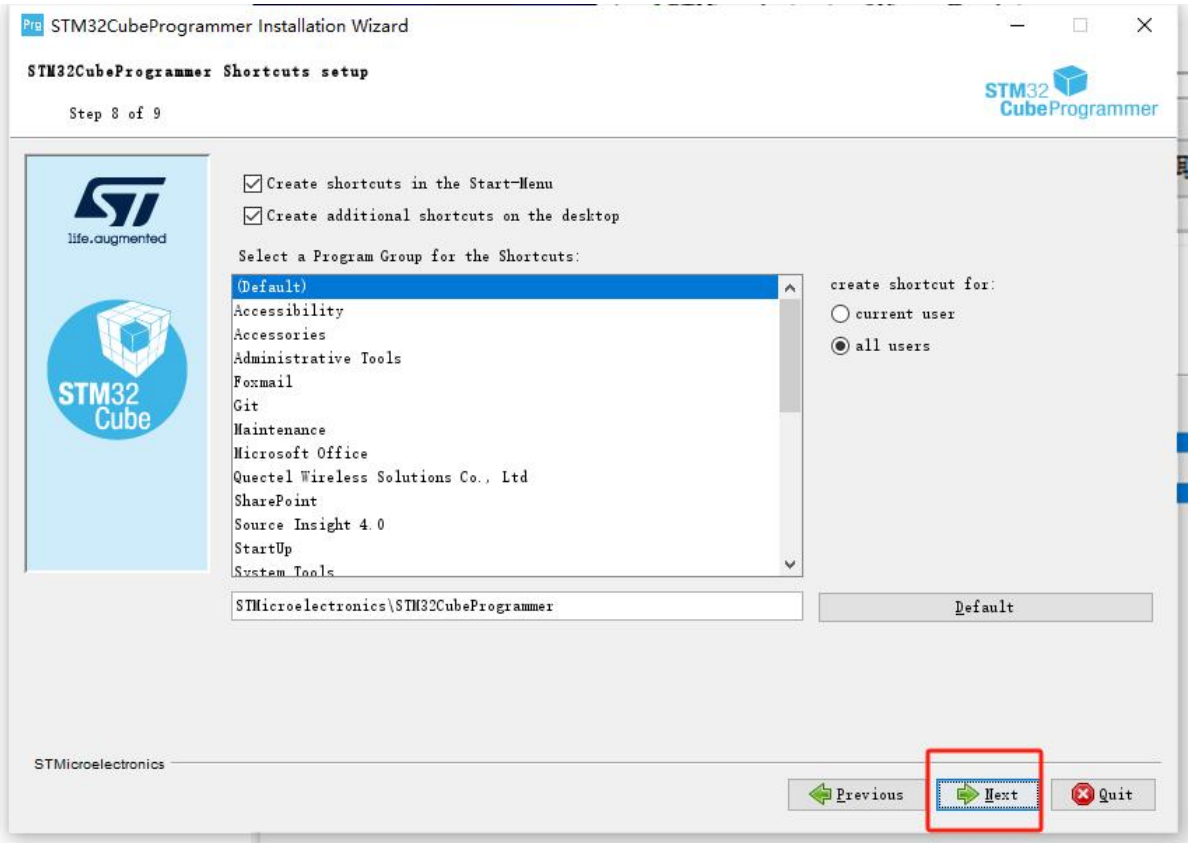


图 2.11 烧录软件安装

安装完成后桌面生成快捷启动方式 

，双击运行。

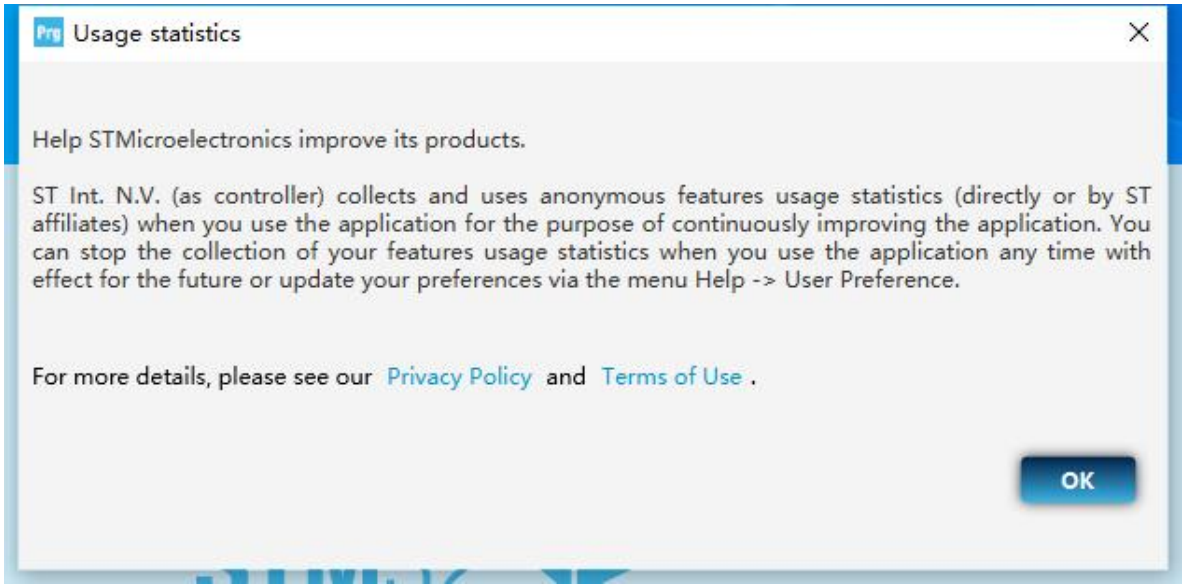


图 2.12 运行 STM32CubeProgrammer

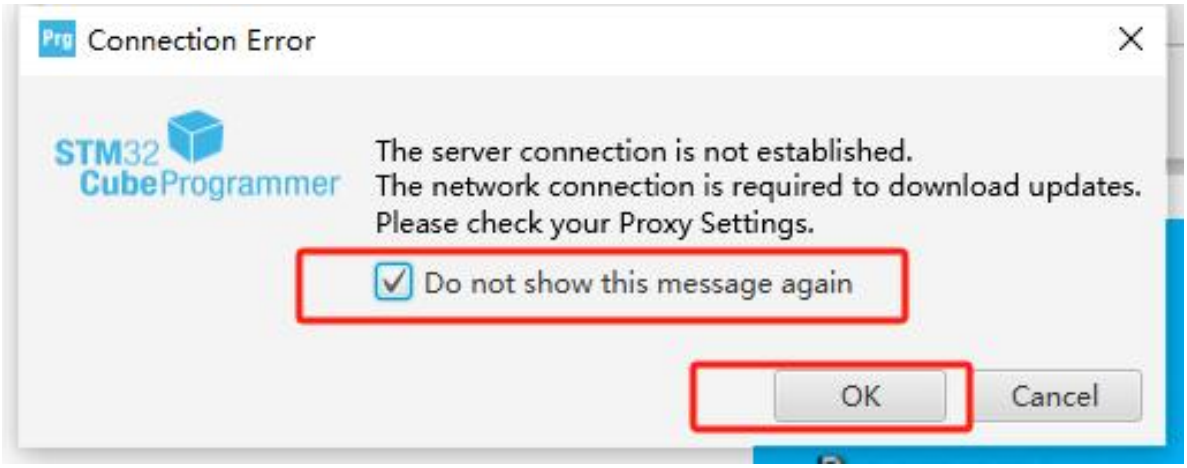
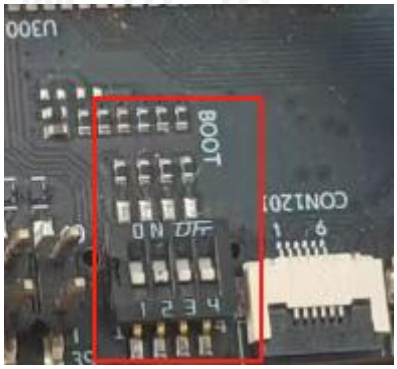


图 2.13 运行 STM32CubeProgrammer

如下 STM32CubeProgrammer 安装完成。

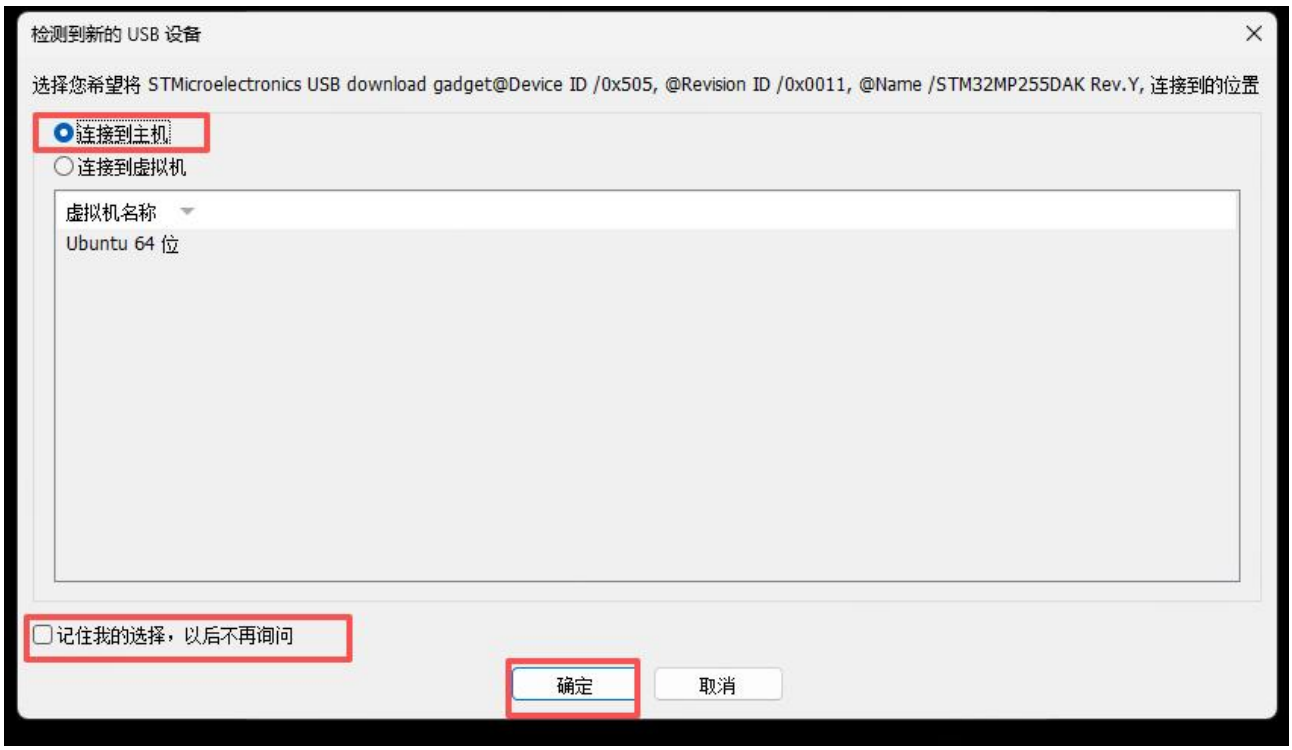
6.2 STM32CubeProgrammer 烧录镜像

如下图，为开发板启动配置引脚，将拨码全部拨到数字侧，如下，则为 otg 烧录启动方式，



然后对开发板重新上电或按复位按键，开发板重新上电后 OTG 座子连接 pc，选择升级方式为 usb，点击刷新按键出现 USB 设备。点击 connect 连接 ic610。如下图：

注意：虚拟机开启时会弹出 usb 选择窗口，此时需要手动选择连接 windows。



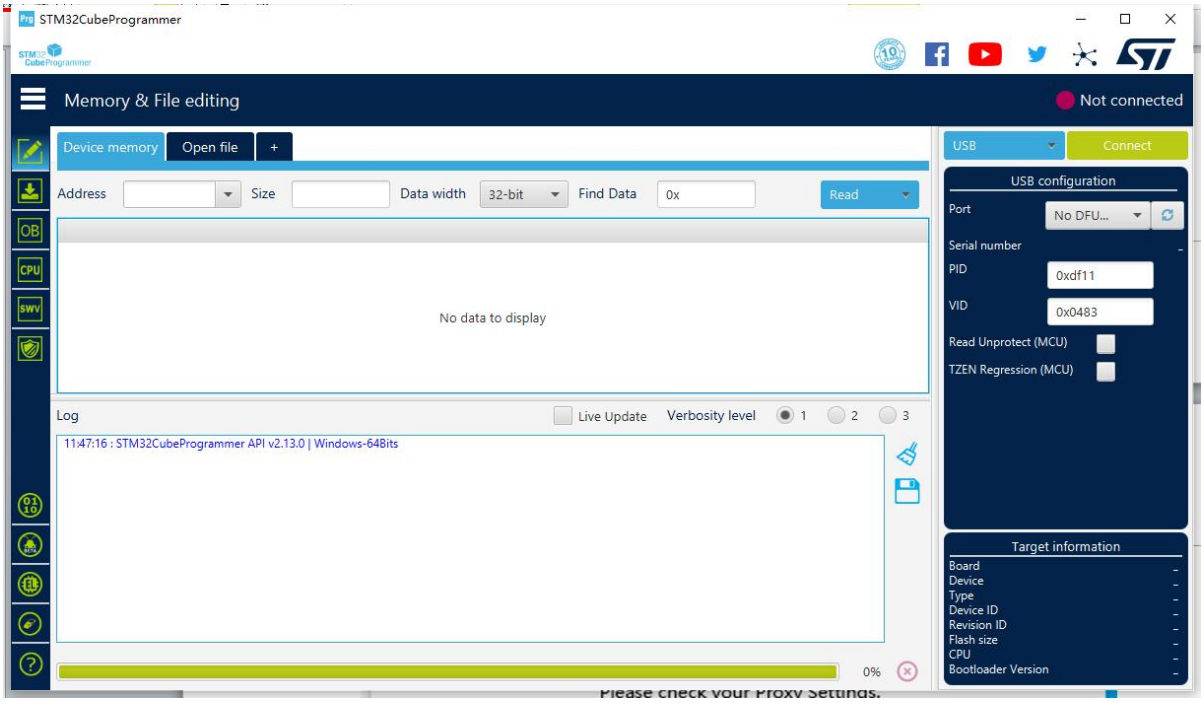
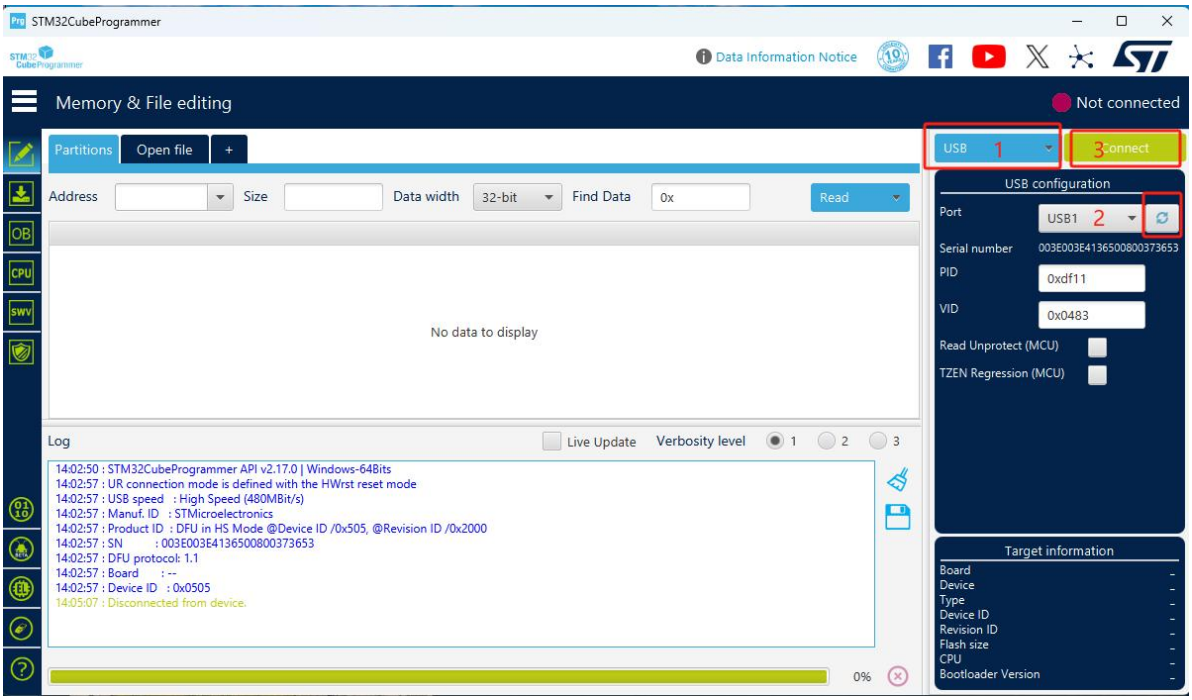
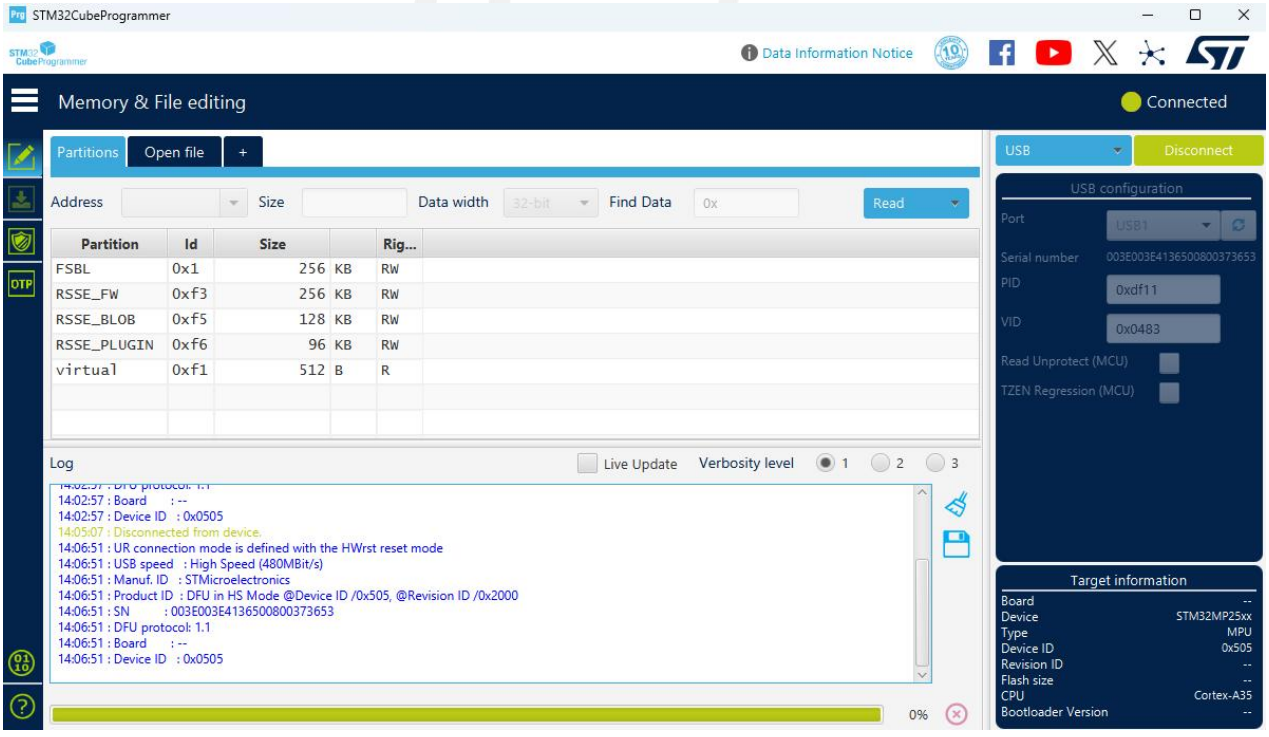


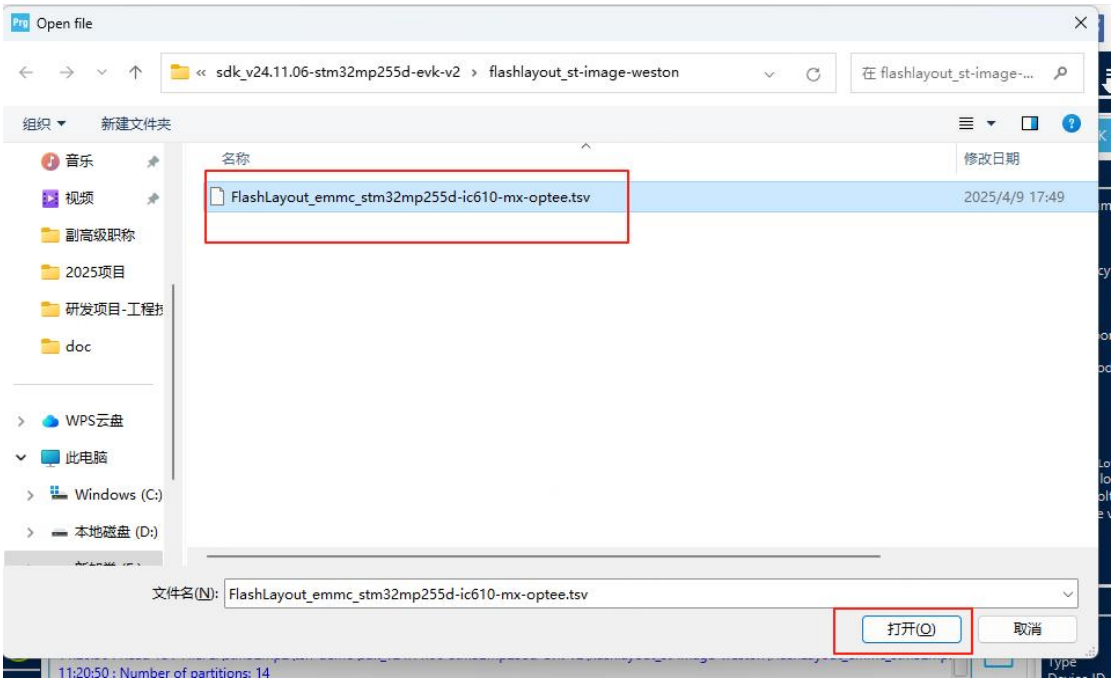
图 2.14 STM32CubeProgrammer 烧录界面



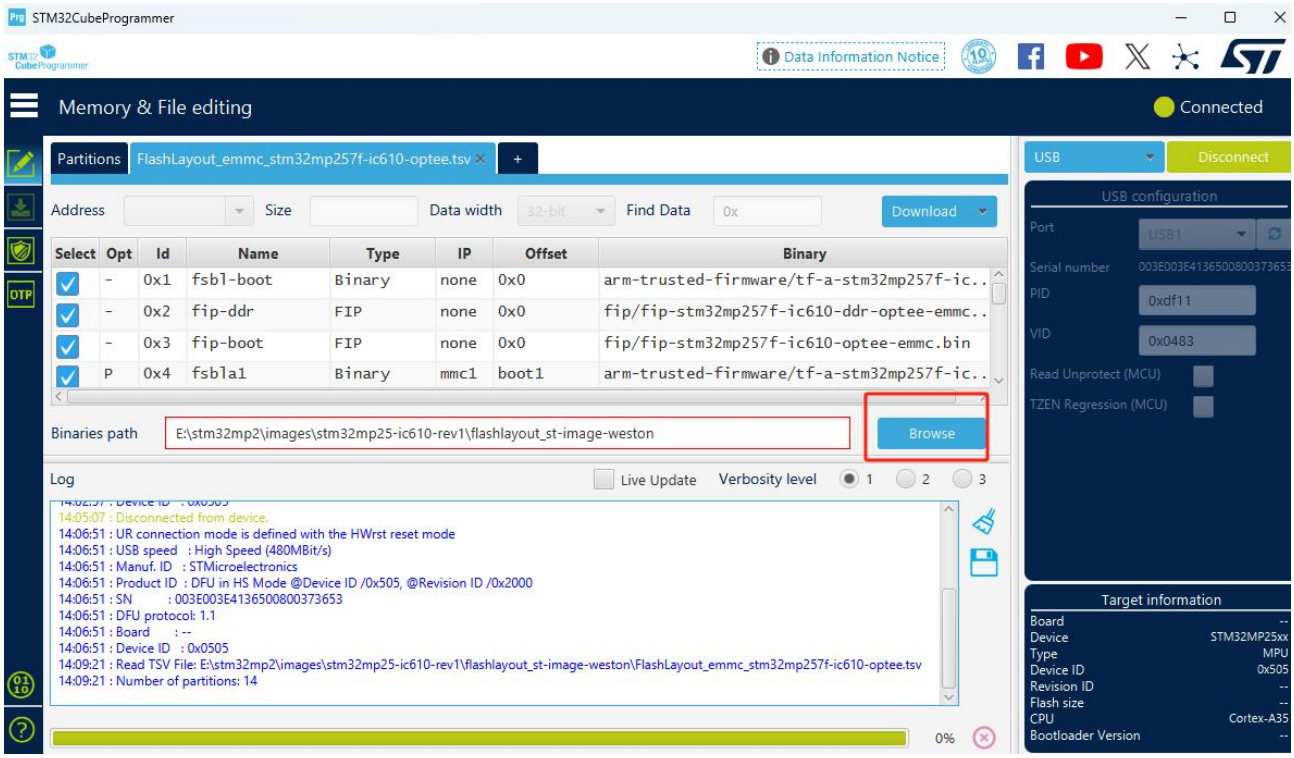
点击“Open file”选择镜像烧录包下的 sdk_v24.11.06-stm32mp255d-evk-v2\flashlayout_st-image-weston\FlashLayout_emmc_stm32mp255d-ic610-mx-optee.tsv

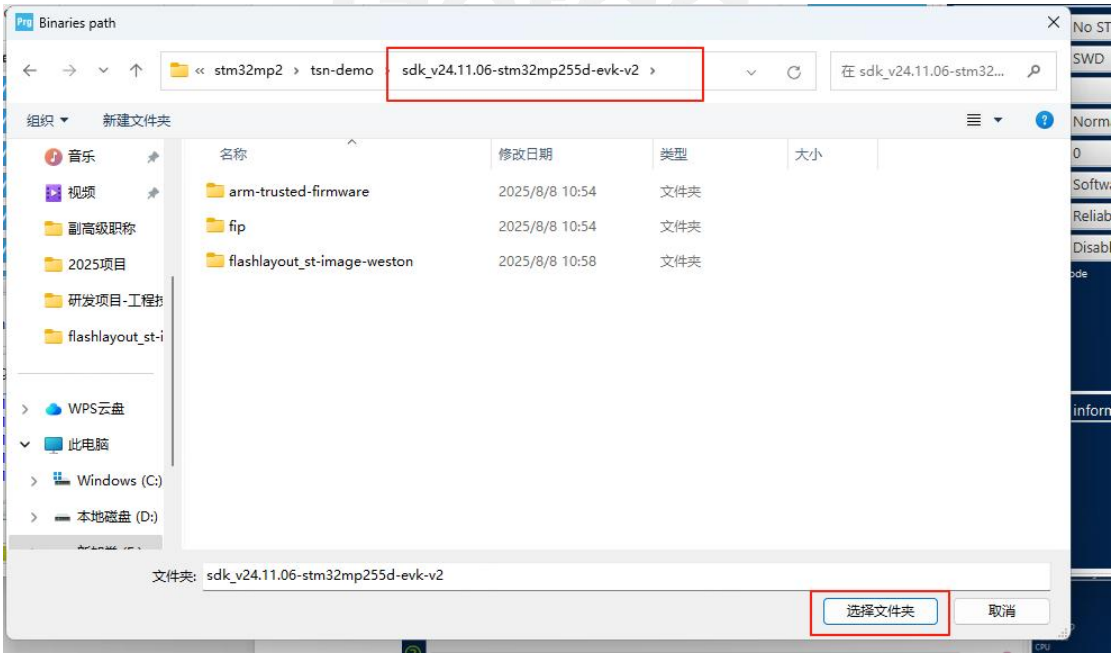


选择 FlashLayout_emmc_stm32mp255d-ic610-mx-optee.tsv 后点击“打开”

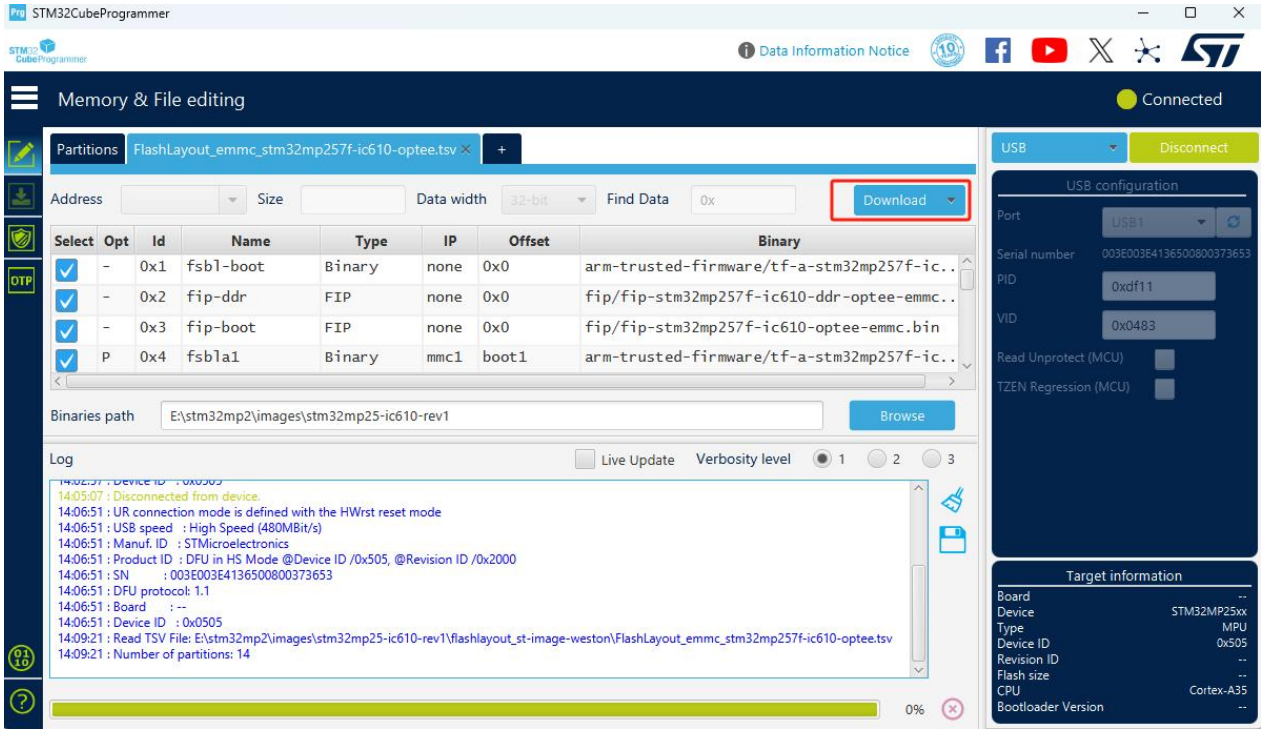

Open file
sdk_v24.11.06-stm32mp255d-evk-v2 › flashlayout_st-image-weston
在 flashlayout_st-image-...
组织
新建文件夹
音乐
视频
副高级职称
2025项目
研发项目-工程
doc
WPS云盘
此电脑
Windows (C:)
本地磁盘 (D:)
名称
修改日期
FlashLayout_emmc_stm32mp255d-ic610-mx-optee.tsv
2025/4/9 17:49
文件名(N):
FlashLayout_emmc_stm32mp255d-ic610-mx-optee.tsv
打开(O)
取消
11:20:50 : Number of partitions: 14

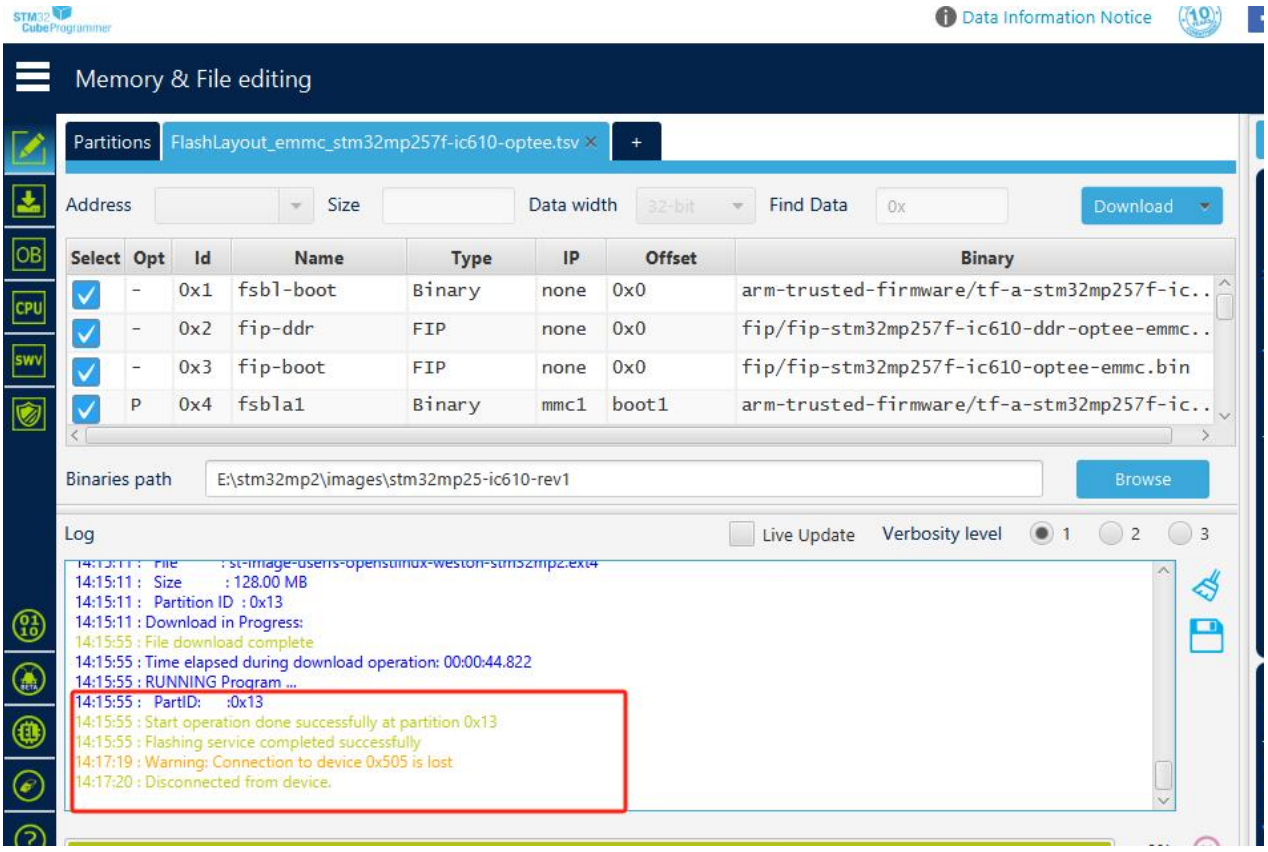
点击“Browse”选择 sdk_v24.11.06-stm32mp255d-evk-v2 所在路径。





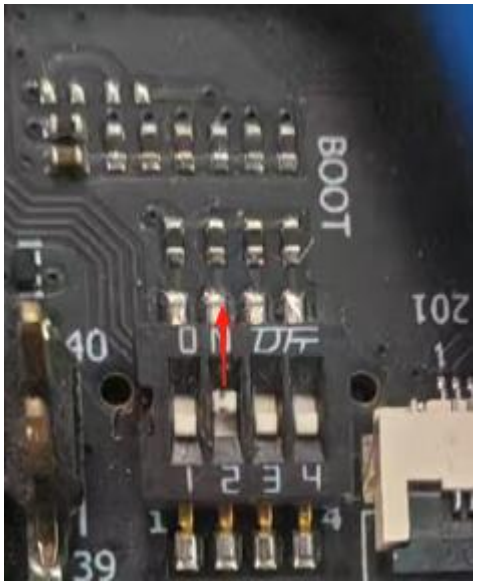
如下配置完成，点击 Dowload 进行下载。





如上图表示下载完成，将拨码拨到 emmc 启动方式，拨码 2 拨到字母 on 侧，重新上电

或按复位按键进行启动。



7 YOCTO_v24.11.06 开发环境搭建

yocto 源码：

```
cd /home/lsd/ic610/stm32mp2_yocto_v24.11.06
```

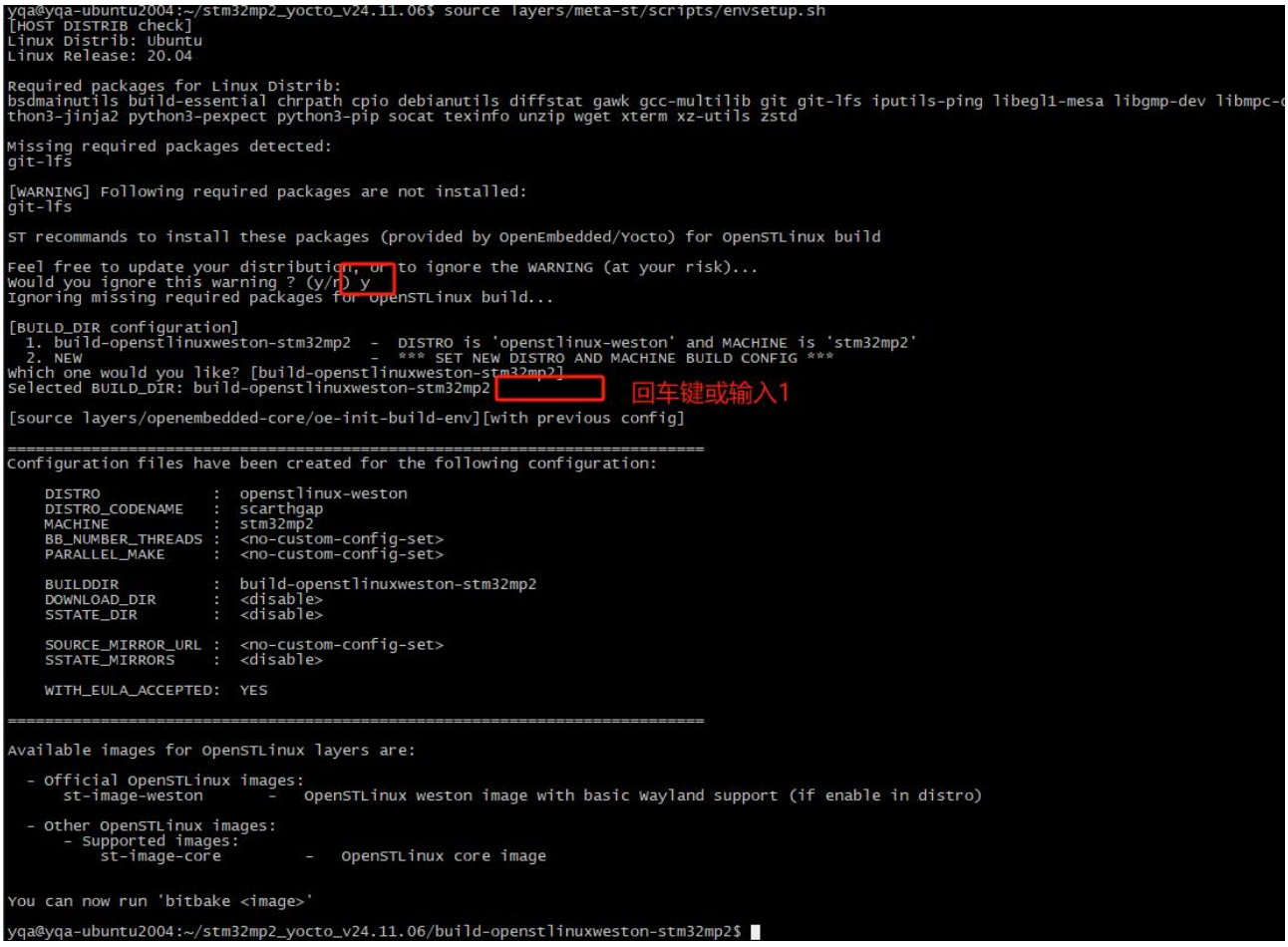
yocto 环境变量配置：

```
stm32mp2_yocto_v24.11.06$ DISTRO=openstlinux-weston MACHINE=stm32mp2
source layers/meta-st/scripts/envsetup.sh
```

配置后再次配置可直接执行：

```
stm32mp2_yocto_v24.11.06$ source layers/meta-st/scripts/envsetup.sh
```

U 盘下提供的 ubuntu2404 虚拟机已经存在完成的 yocto 源码，故使用此虚拟机，直接使用 source layers/meta-st/scripts/envsetup.sh 配置 yocto 环保变量后直接编译即可。



注意：

在新的终端中必须再次执行 source layers/meta-st/scripts/envsetup.sh

编译 weston 镜像：

```
build-openstlinuxweston-stm32mp2$ bitbake st-image-weston
```

编译完成后 tmp-glibc/deploy/images/stm32mp2/下生成全部镜像：

st-image-weston-openstlinux-weston-stm32mp2.rootfs.ext4 为文件系统

st-image-userfs-openstlinux-weston-stm32mp2.userfs.ext4

st-image-vendorfs-openstlinux-weston-stm32mp2.vendorfs.ext4

生成交叉编译器：

```
build-openstlinuxweston-stm32mp2$ bitbake -c populate_sdk st-image-weston
```

生成交叉编译器安装包：

tmp-glibc/deploy/sdk/meta-toolchain-openstlinux-weston-stm32mp2.rootfs-x86_64-toolchain-5.0.3-snapshot.sh

7.1 TSN 开发环境搭建

7.1.1 X-LINUX-TSNSWCH 扩展包

YoctoX-LINUX-TSNSWCH 扩展包是一个 Yocto 层，提供用户空间和 Linux 内核层级所需的资源，用于管理时间敏感网络和硬件以太网交换机。该扩展包可简化开发流程，以实现：

1、配置硬件以太网交换机并应用 TSN 配置方案；

2、配置硬件以太网交换机（又称 ETHSW 内部外设）但不采用 TSN 配置方案；

3、提供配置 TSN 端点所需的必备工具集。

默认情况下，将此扩展包添加至 Linux 发行版后，硬件以太网交换机即可直接使用，同时会安装 TSN 配置工具。

7.1.2 Yocto 下编译 X-LINUX-TSNSWCH

下载 tsn 源码

```
git config --global http.postBuffer 209715200
git config --global https.postBuffer 209715200
cd tm32mp2_yocto_v24.11.06/layers/meta-st
git clone -b scarthgap https://github.com/STMicroelectronics/meta-st-stm32mp-tsn-swch.git
git clone -b scarthgap https://github.com/STMicroelectronics/meta-st-stm32mp-tsn-acm.git
cd ../..
DISTRO=openstlinux-weston MACHINE=stm32mp2 source layers/meta-st/scripts/envsetup.sh
bitbake-layers add-layer ../layers/meta-st/meta-st-stm32mp-tsn-swch
bitbake-layers add-layer ../layers/meta-st/meta-st-stm32mp-tsn-acm
bitbake -c populate_sdk st-image-weston
```

编译完成后 tsn 相关工具及库文件添加到 tmp-glibc/deploy/images/stm32mp2/st-image-weston-openstlinux-weston-stm32mp2.rootfs.ext4

7.1.3 Kernel 使能 tsn 相关功能

在 kernel 源码路径下：

```
linux-6.6.48$ source /opt/st/stm32mp2/5.0.3-openstlinux-6.6-yocto-scarthgap-mpu-v24.11.06/environment-setup-cortexa35-ostl-linux
linux-6.6.48$ make menuconfig
```

使能以下配置项：

[*] Networking support --->

Networking options --->

[*] QoS and/or fair queueing --->

<*> Credit Based Shaper (CBS)

<*> Time Aware Priority (taprio) Scheduler

<*> Multi-queue priority scheduler (MQPRIO)

[*] Actions --->

<*> Traffic Policing

<*> Generic actions

<*> Redirecting and Mirroring

<*> SKB Editing

<*> Vlan manipulation

<*> Frame gate entry list control tc action

IC610 SDK kernel 下已经默认使能上述配置项，此次培训无需重新配置及编译。

8 IC610 TSN

8.1 IC610 TSN 硬件接口资源

IC610 核心板 CPU 为 stm32mp255d，支持 2 路千兆以太网，且 2 路以太网均支持 tsn 功能。

8.2 TSN 网络基本功能介绍

TSN（时间敏感网络） 是一系列 IEEE 标准技术的集合，它建立在标准以太网之上，旨在为其提供确定性通信的能力。让同一个以太网网络上，关键的高优先级流量（如机器控制信号、自动驾驶指令、机器人同步数据）和普通的低优先级流量（如文件传输、视频监控）可以共存且互不干扰。高优先级流量总能获得它所需的网络资源，并保证极低的延迟和抖动。

8.2.1 时钟同步

IEEE 1588 协议是一个精密时间协议（PTP），用于同步计算机网络中的时钟。 在局域网中，它能将时钟精确度控制在**亚微秒**范围内，为网络中各设备提供统一的时间基准，确保通信的确定性，使其适用于测量和控制系统。

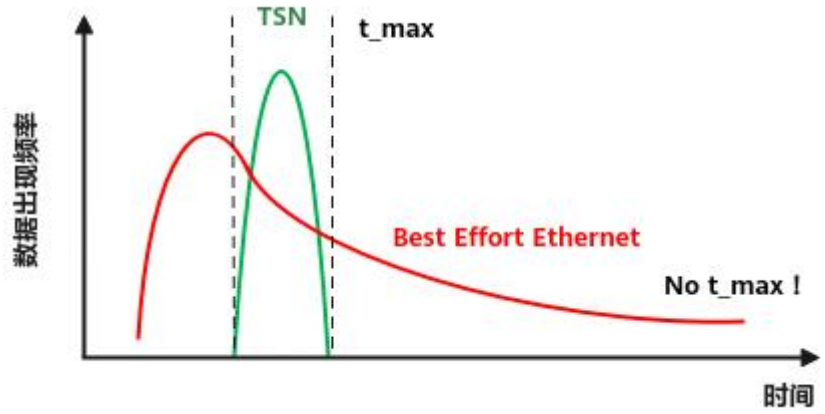
使用场景介绍：

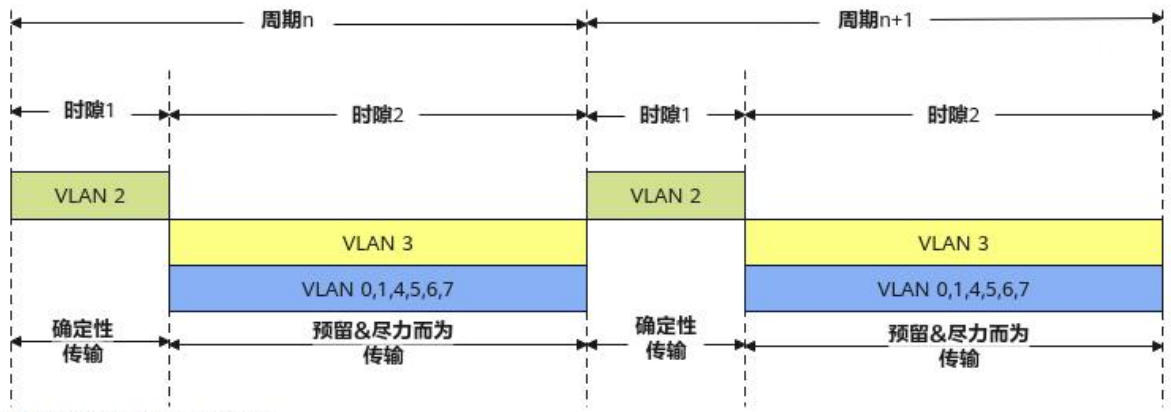
在工业自动化（如机器人协同操作、运动控制）和汽车网络（如线控驱动）。

在分布式系统中，确定一个事件（如传感器检测到物体、执行器到达某个位置）发生的精确时间至关重要，用于数据融合、故障诊断和系统分析。

8.2.2 数据调度及流量整形

TSN 用于数据调度和流量整形的协议有 IEEE 802.1Qav、IEEE 802.1Qbv、IEEE 802.1Qbu、IEEE 802.1Qch 及 IEEE 802.1Qcr。其中，IEEE 802.1Qbv 采用非抢占式的的数据调度，流量调度方式通过时隙进行控制，需要实时传输的数据流优先传输，同时为 best-effort 数据及预留数据预留带宽，允许时间敏感流和非时间敏感流在同一个网络中传输，并确保数据的实时传输。



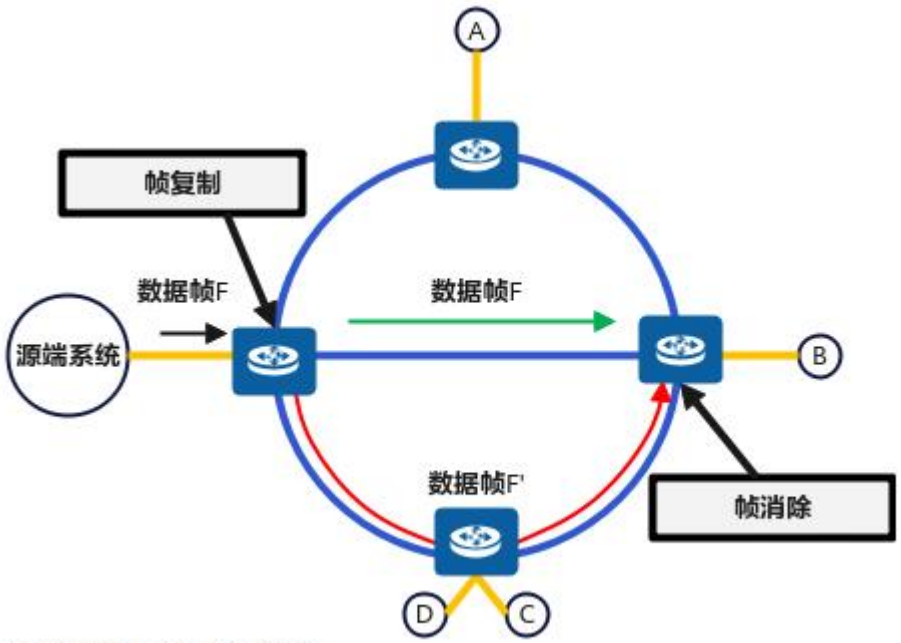


IEEE 802.1Qbv时隙划分图

TSN 通过数据调度及流量整形，能够精确控制数据包的传输时间，确保关键数据在规定时间内到达目的地。同时，还能根据不同业务的需求，精确分配带宽资源，提高带宽利用效率，避免低优先级流量对高优先级流量的干扰，保障关键业务的正常运行，广泛应用于工业自动化、汽车电子、自动驾驶等对实时性和可靠性要求极高的领域。

8.2.3 高可靠性与无缝冗余

对数据传输实时性要求高的应用除了需要保证数据传输的时效性，同时也需要高可靠的数据传输机制，以便应对网桥节点失效、线路断路和外部攻击带来的各种问题，来确保功能安全和网络安全。IEEE 802.1Qci、IEEE 802.1CB 及 IEEE 802.1Qca 用于实现 TSN 这方面的性能。



IEEE 802.1 CB工作原理

IEEE 802.1CB 为以太网提供双链冗余特性，通过在网络的源端系统和中继系统中对每个数据帧进行序列编号和复制，并在目标端系统和其他中继系统中消除这些复制帧，确保仅有一份数据帧被接收。可用来防止由于拥塞导致的丢包情况，也可以降低由于设备故障造成分

组丢失的概率及故障恢复时间，提高网络可靠性。

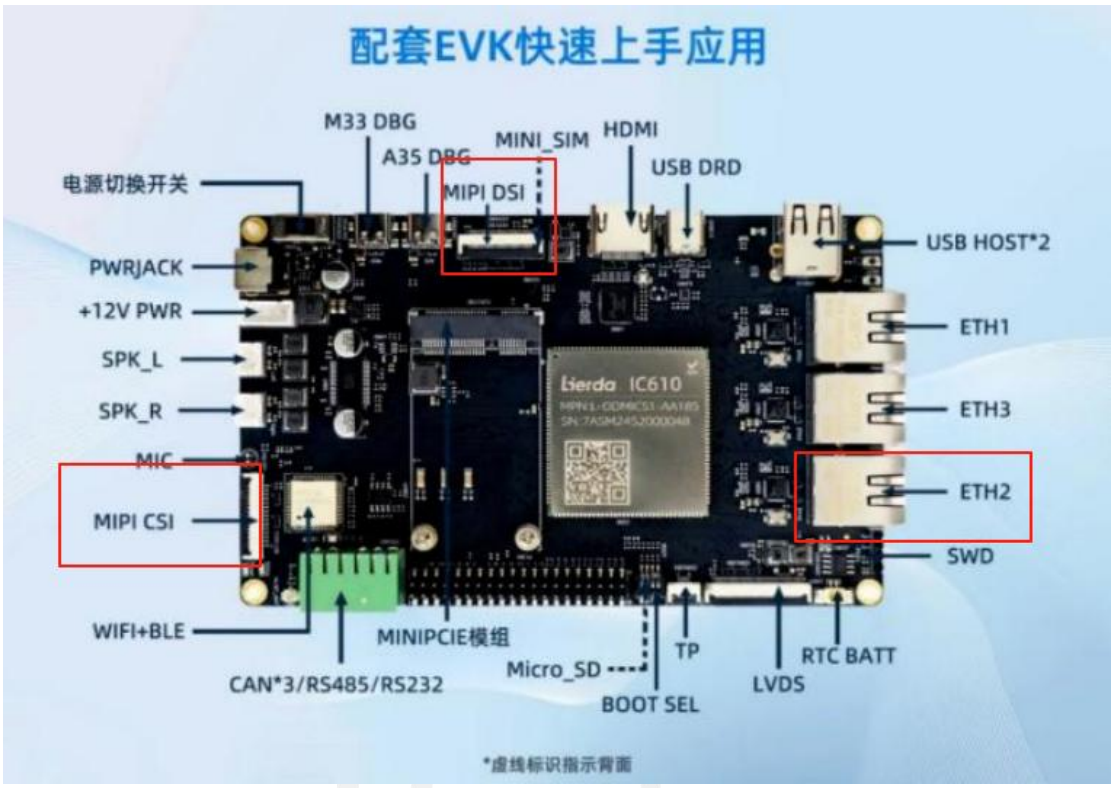
8.2.4 资源管理

TSN 资源管理子协议包括 IEEE 802.1Qat 协议和 IEEE 802.1Qcc 协议。IEEE 802.1Qcc 协议是 IEEE802.1Qat 协议的增强。

IEEE802.1Qat 即流预留协议。根据流的资源要求和可用的网络资源情况指定数据准入控制，保留资源并通告从数据源发送端至数据接收端之间的所有网络节点，确保指定流在整条传输路径上有充足的网络资源可用。

IEEE 802.1Qcc 则进一步扩展为集中式和分布式资源配置方式，使带宽分配更加灵活高效。该机制可确保关键流量的性能不受其他业务干扰，实现业务级别的服务保障。

8.3 IC610 TSN PPS demo



2 个板子烧录完镜像启动后，type-c 线连接 A35 DBG,pc 端查看端口号，secureCRT 或其他软件打开串口。2 个板子通过 ETH2 连接网络。

板子 1:

```
ifconfig end0 192.168.1.10

ptp4l -m -i end0    &

date -s "2025-9-19 11:11:11"

date
```

板子 2:

```
Ifconfig end0 192.168.1.11

ptp4l -q 1 -i end0    -s &
```

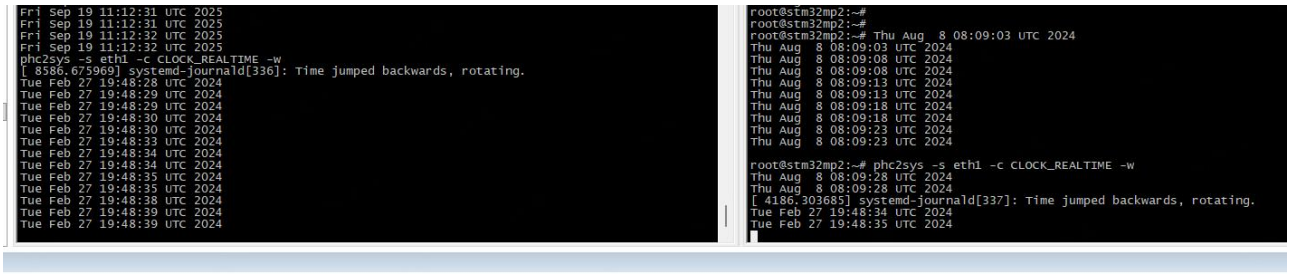
```
date -s "2024-8-8 8:8:8"

date
```

2 个板子端运行：

```
phc2sys -s end0 -c CLOCK_REALTIME -w
```

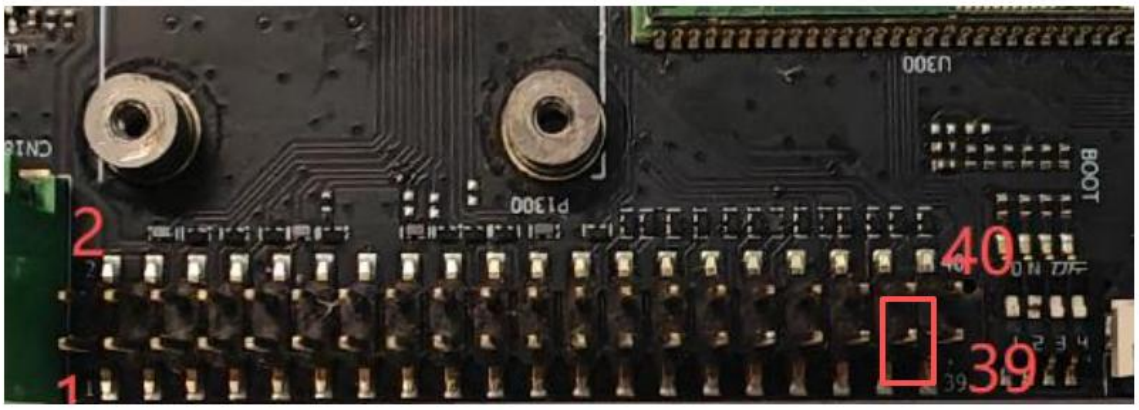
通过 date 查询系统时间

时间自动同步：



网口支持 pps 输出，配置 gpio 为 pps 输出模式，2 个板子输出的 pps，可通过示波器查看波形相位差来查看时间误差，测试方式如下：

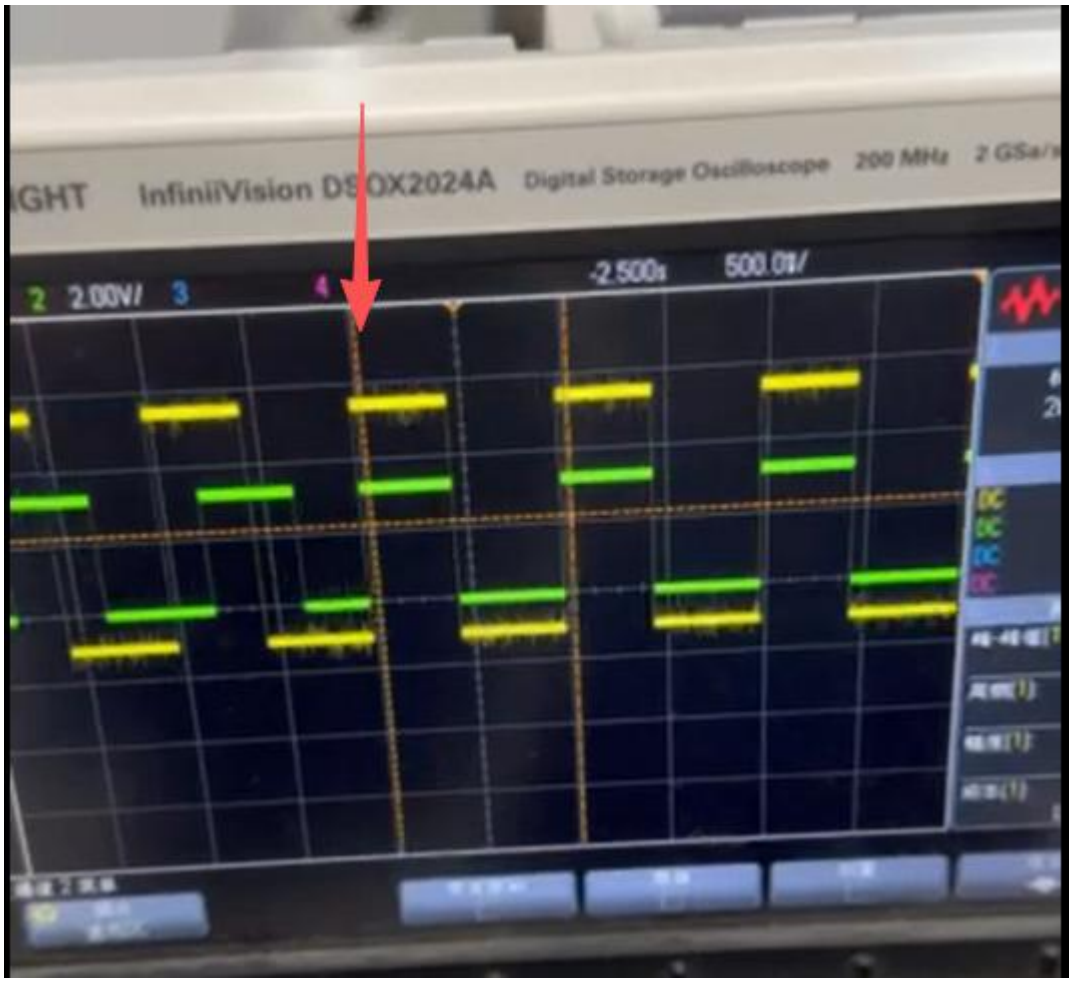
37 引脚为 pps 输出引脚，示波器同时测量 2 个板子波形。



时间未同步前：


KEYSIGHT
InfiniiVision DSOX2024A
Digital Storage Oscilloscope
200 MHz
2 GSa/s
MEGA
-2.500s
500.0ns/

时间同步中：


InfiniiVision DSOX2024A
Digital Storage Oscilloscope
200 MHz
2 GSa/s
2 2.00V/ 3
4
-2.500s
500.0ns/

时间同步结束：

以下为视频效果：ST-LIERDA_IC610 高阶实战培训资料\video\ptp 时钟同步.mp4

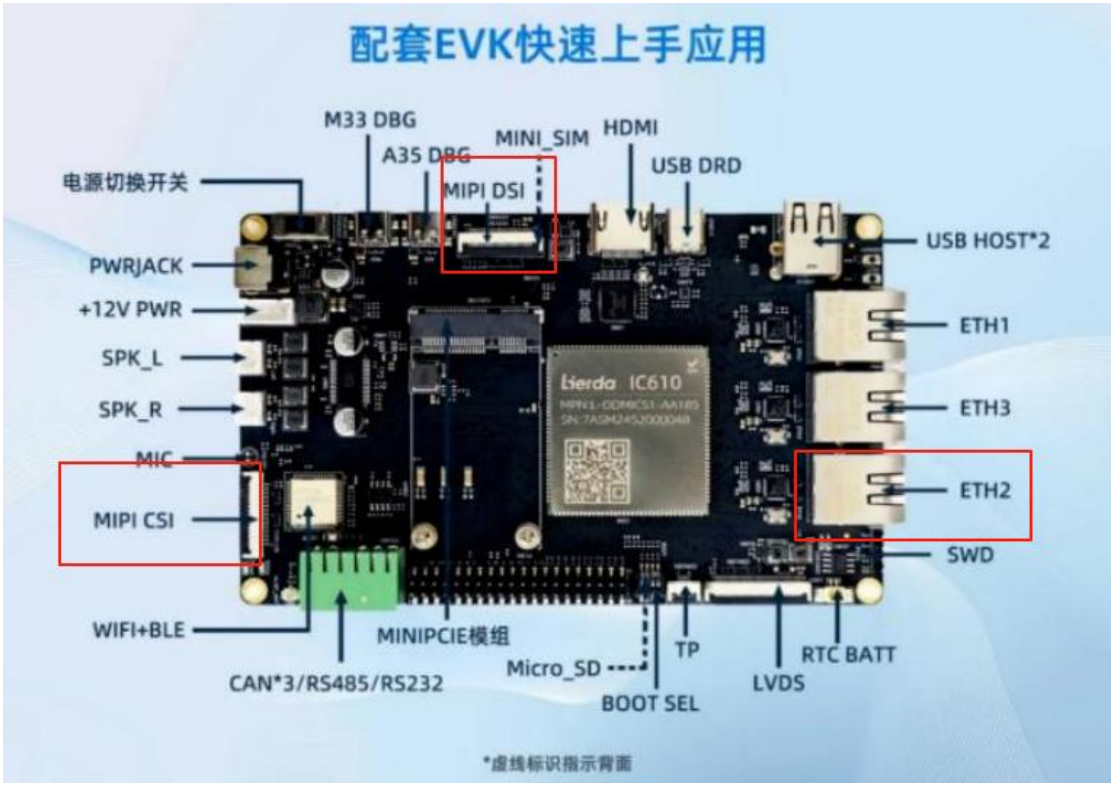
8.4 IC610 TSN RTSP demo

8.4.1 硬件运行环境

1、发送板： 连接 mipi 屏幕（MIPI DSI ）、摄像头(MIPI CSI)，网线连接以太网网口 ETH2

2、接收板： 连接 mipi 屏幕（MIPI DSI ）、摄像头(MIPI CSI)，网线连接以太网网口 ETH2

如下图：



2 个板子烧录完镜像启动后，type-c 线连接 A35 DBG,pc 端查看端口号，secure crt 或其他软件打开串口。

8.4.2 接收板串口

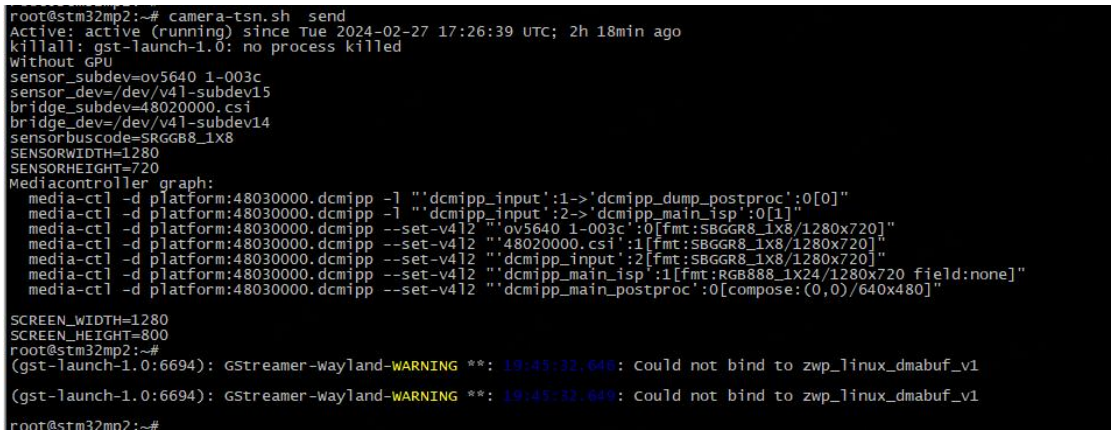
```
root@stm32mp2:~# camera-tsn.sh receive
```

```
root@stm32mp2:~# camera-tsn.sh receive
Active: active (running) since Tue 2024-02-27 17:26:38 UTC; 12min ago
killall: iperf3: no process killed
killall: gst-launch-1.0: no process killed
Setting pipeline to PAUSED ...

(gst-launch-1.0:2328): GStreamer-Wayland-WARNING **: 17:39:27.866: Could not bind to zwp_linux_dmabuf_v1

(gst-launch-1.0:2328): GStreamer-Wayland-WARNING **: 17:39:27.869: Could not bind to zwp_linux_dmabuf_v1
Pipeline is live and does not need PREROLL ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Redistribute latency...
Redistribute latency...
Redistribute latency...
Redistribute latency...
0:00:03.4 / 99:99:99.
```

8.4.3 发送板串口

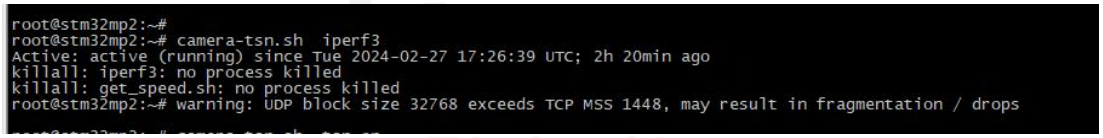
```
root@stm32mp2:~# camera-tsn.sh send
```

开启摄像头到屏幕及推流视频到网口，观察接收端屏幕视频流，接收端视频正常



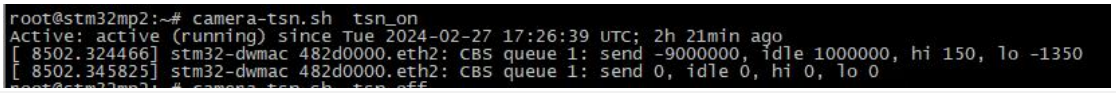
root@stm32mp2:~# camera-tsn.sh iperf3

开启 iperf3 ，旋转摄像头以改变摄像头画面，接收端视频延时增大、卡顿、花屏



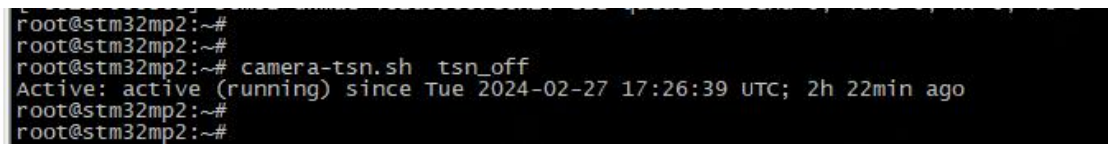
root@stm32mp2:~# camera-tsn.sh tsn_on

开启 tsn cbs 接收端屏幕视频流恢复流畅



root@stm32mp2:~# camera-tsn.sh tsn_off

关闭 tsn cbs 接收端屏幕视频延时增大、卡顿



```
root@stm32mp2:~# camera-tsn.sh tsn_on
```

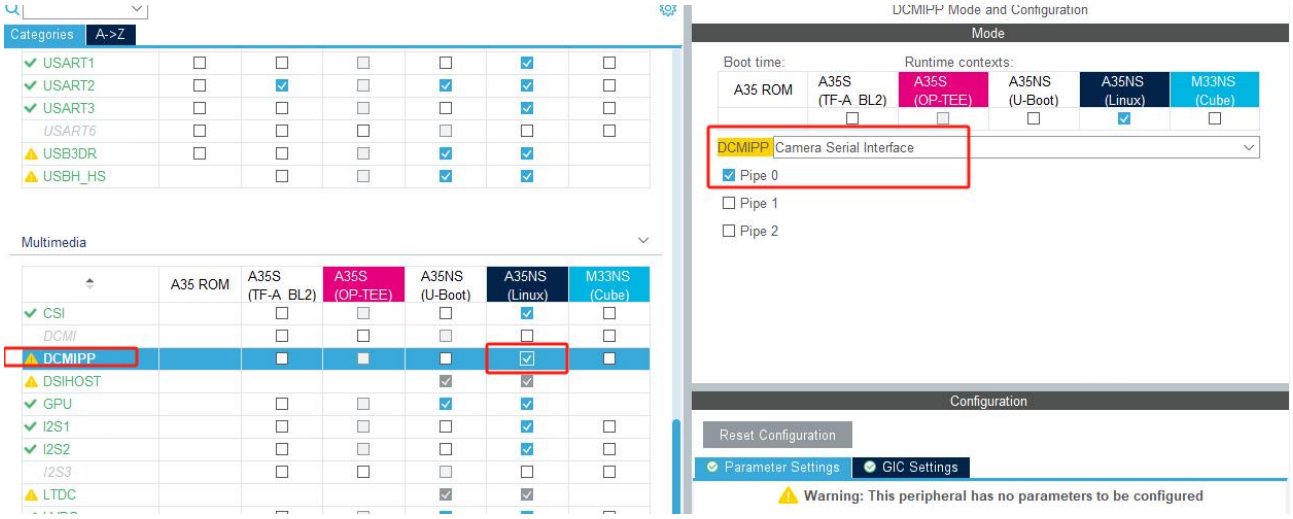
再次开启 tsn cbs 接收端屏幕视频流恢复流畅

可播放视频：ST-LIERDA_IC610 高阶实战培训资料\video\tsn 效果展示.mp4

9 IC610 MIPI CSI 模块移植

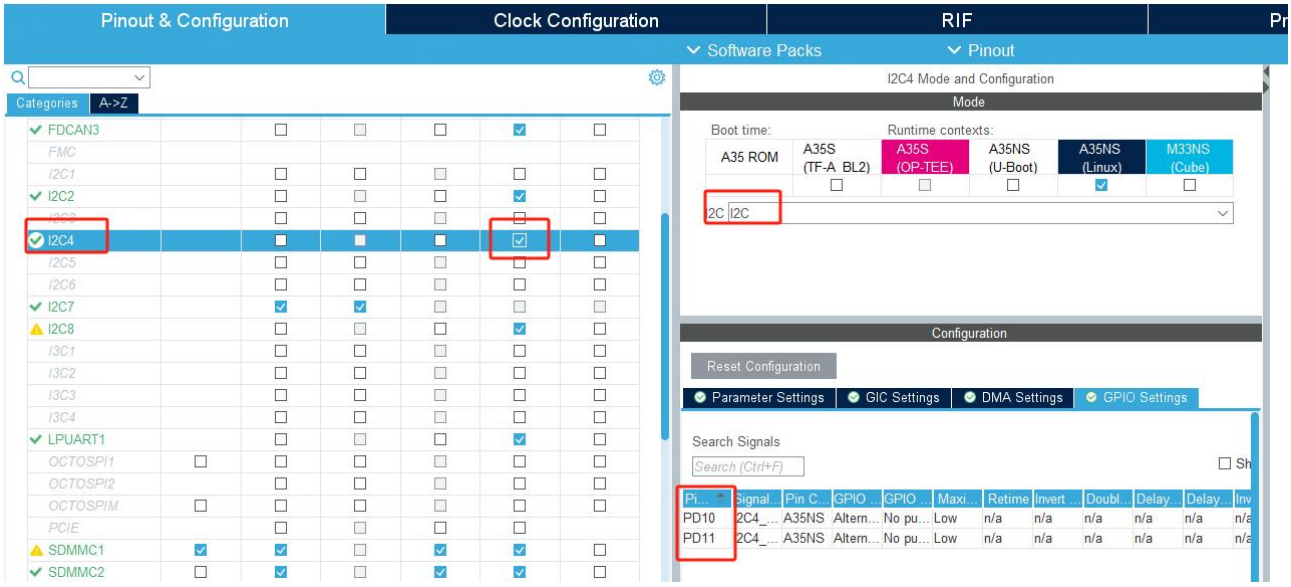
9.1 摄像头驱动开发

Cubemx6.14 打开 ic610.ioc，本 evk 的摄像头为 ov5645，故需要先使能 csi 总线驱动。

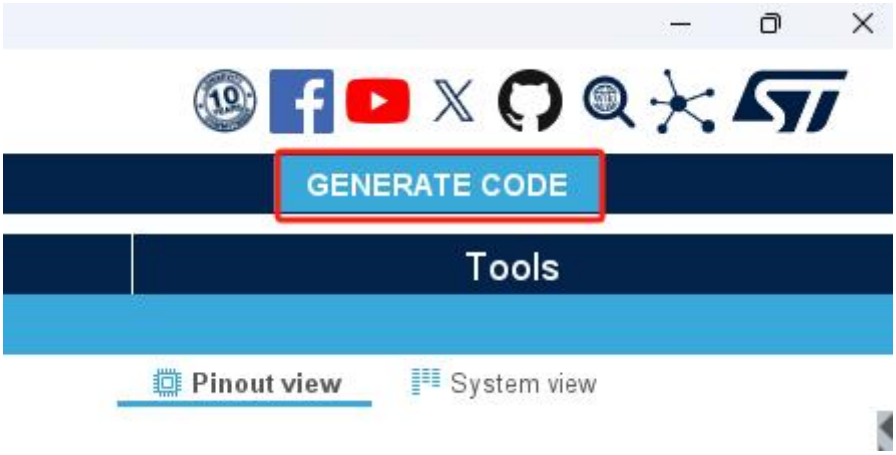


Csi 总线驱动默认使用上述配置即可，其他设置本 sdk 暂不支持。

摄像头控制引脚为 i2c，根据实际硬件设计使能 i2c 总线。如本 evk 连接摄像头的 i2c 为 i2c4，使用 gpio 为 PD10、PD11,cubemx 端使能 i2c4。



上述配置完成后点击 GENERATE CODE，即可更新 dts。



代码生成完成后，自动更新

ic610\CA35\DeviceTree\ic610\kernel\stm32mp255d-ic610-mx.dts

生成 csi 和 dcmipp 及 i2c4 节点

在以下区域内添加相关配置信息即可。

```
&csi {
status = "okay";
```

```
/* USER CODE BEGIN csi */
vdd-supply =    <&scmi_vddcore>;
vdda18-supply = <&scmi_v1v8>;
ports {
 #address-cells = <1>;
 #size-cells = <0>;
 port@0 {
  reg = <0>;
  csi_sink: endpoint {
   remote-endpoint = <&ov5645_ep>;
   data-lanes = <0 1>;
   bus-type = <4>;
  };
 };
 port@1 {
  reg = <1>;
  csi_source: endpoint {
   remote-endpoint = <&dcmipp_0>;
  };
```

```
  };
 };
 /* USER CODE END csi */
};

&dcmipp {
 status = "okay";

 /* USER CODE BEGIN dcmipp */
 port {
  dcmipp_0: endpoint {
   remote-endpoint = <&csi_source>;
   bus-type = <4>;
  };
 };
 /* USER CODE END dcmipp */
};
```

```
&i2c4 {
 pinctrl-names = "default", "sleep";
 pinctrl-0 = <&i2c4_pins_mx>;
 pinctrl-1 = <&i2c4_sleep_pins_mx>;
 status = "okay";


 /* USER CODE BEGIN i2c4 */
  ov5640: ov5640@3c {
                                compatible = "ovti,ov5640";
                                reg = <0x3c>;
                        status = "okay";
                        clocks = <&clk_ext_camera>;
                                clock-names = "xclk";
                                clock-frequency = <24000000>;


                             DOVDD-supply = <&ov5640_vdddo_1v8>;
                              AVDD-supply = <&ov5640_vdda_2v8>;
                            DVDD-supply = <&ov5640_vddd_1v5>;
                           rotation = <180>;
```

```
                        port {
                                ov5640_ep: endpoint {

                                        clock-lanes = <0>;
                                        data-lanes = <0 1>;
        remote-endpoint = <&csi_sink>;
    link-frequencies = /bits/ 64 <594000000>;
                                };
                        };
                };
 /* USER CODE END i2c4 */
};
```

对于不同摄像头，执行修改 csi 下 remote-endpoint = <&ov5640_ep>；即可。

同时 kernel 下使能摄像头驱动 CONFIG_VIDEO_OV5640，重新编译全部镜像并更新到开发板中。

9.2 用户层操作

摄像头预览,可实时输出摄像头内容到屏幕:

```
/usr/local/demo/application/camera/bin/launch_camera_preview_mp25.sh
```

添加其他摄像头可修改如下脚本:

```
/usr/local/demo/application/camera/bin/launch_camera_control_mp25.sh
```

如下增加 ov5645

```
case "$sensor_subdev" in
        *"ov5640"*)
            #OV5640 only support 720p with raw-bayer format
            SENSORWIDTH=1280
            SENSORHEIGHT=720
            #OV5640 claims to support all raw bayer combinations but always output SBGGR8_1X8...
            sensorbuscode=SBGGR8_1X8
            ;;
    *"ov5645"*)
            SENSORWIDTH=1280
          SENSORHEIGHT=720
```

```
                sensorbuscode=YUYV8_2X8

            *"imx335"*)

                #IMX335 expose both RGGB10 and RGGB12 however only RGGB10 can
work on CSI 2 lanes

                sensorbuscode=SRGGB10_1X10

                main_postproc=$(media-ctl -d $DCMIPP_MEDIA -e
dcmipp_main_postproc)


                #Enable gamma correction

                v4l2-ctl -d "$main_postproc" -c gamma_correction=1

                #Do exposure correction continuously in background

                sleep 3  && while : ; do /usr/local/demo/bin/dcmipp-isp-ctrl -i0 -g >
/dev/null ; done &

                ;;


                main_postproc=$(media-ctl -d $DCMIPP_MEDIA -e
dcmipp_main_postproc)


                #Enable gamma correction

                v4l2-ctl -d "$main_postproc" -c gamma_correction=1
```

```
                    #Do exposure correction continuously in background
                    sleep 3    && while : ; do /usr/local/demo/bin/dcmipp-isp-ctrl -i0 -g >
/dev/null ; done &
                    ;;
```

10 IC610 AI

10.1 YOCTO 下 AI 开发环境搭建

10.1.1 X-LINUX-AI 扩展包

X-LINUX-AI 是一款免费的开源软件包，专为人工智能领域打造。

它是一个完整的生态系统，基于 OpenSTLinux 的开发人员能够非常轻松地创建人工智能应用程序。具有一下特点：

1、STM32 MPU 开发板一站式 AI 解决方案

2、预集成至基于 ST 环境的 Linux 发行版

3、包含用于执行神经网络模型的 AI 框架

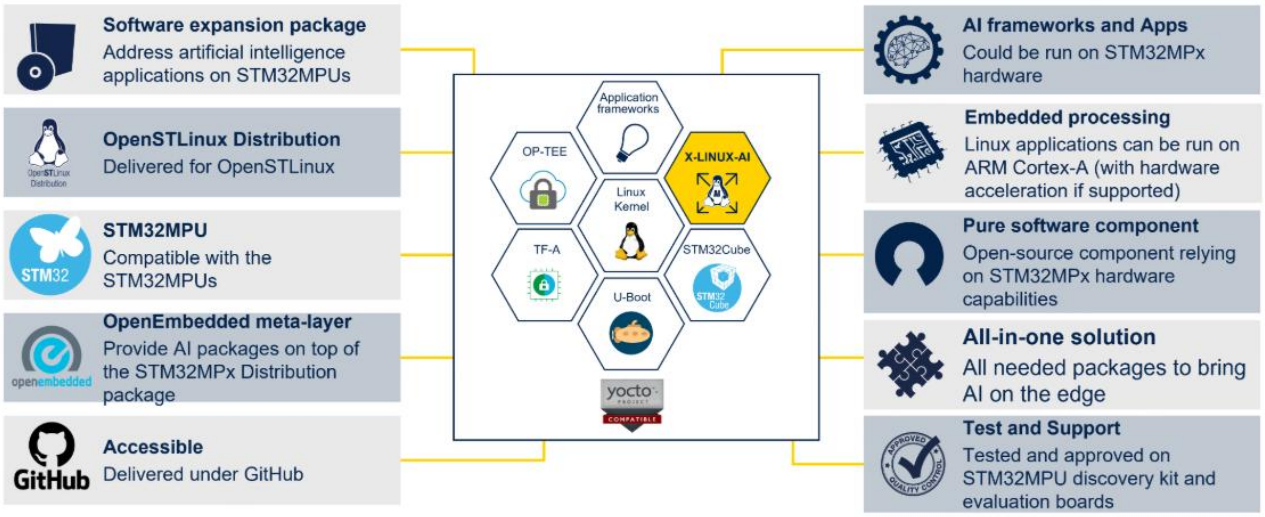
4、提供适用于 MPU 的 AI 模型基准测试应用工具

5、支持通过 Python 语言及 AI 框架的 Python API 快速实现应用原型设计

6、提供 C++ API 以满足嵌入式高性能应用需求

7、提供经过优化的开源解决方案及源代码，支持大量代码复用并节省开发时间

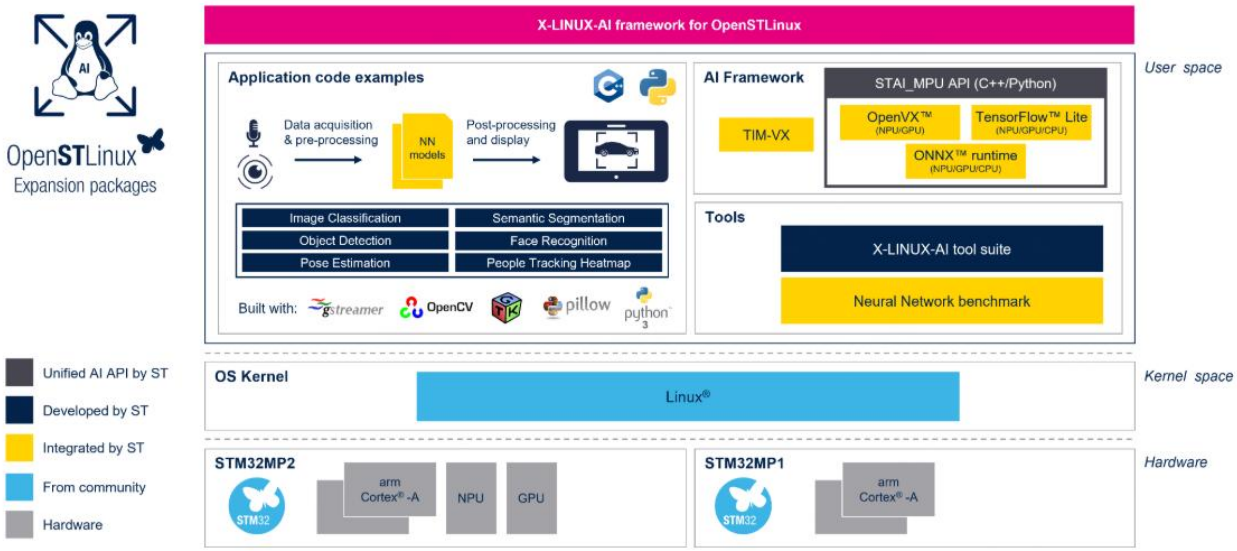
8、通过 GitHub 以 Yocto meta-layer 形式交付



根据 STM32MPU 系列的不同，X-LINUX-AI 支持在神经网络处理单元（NPU）、图形处理单元（GPU）或 Cortex-A 内核（支持多线程）上执行推理运算。不同系列 MPU 如下：

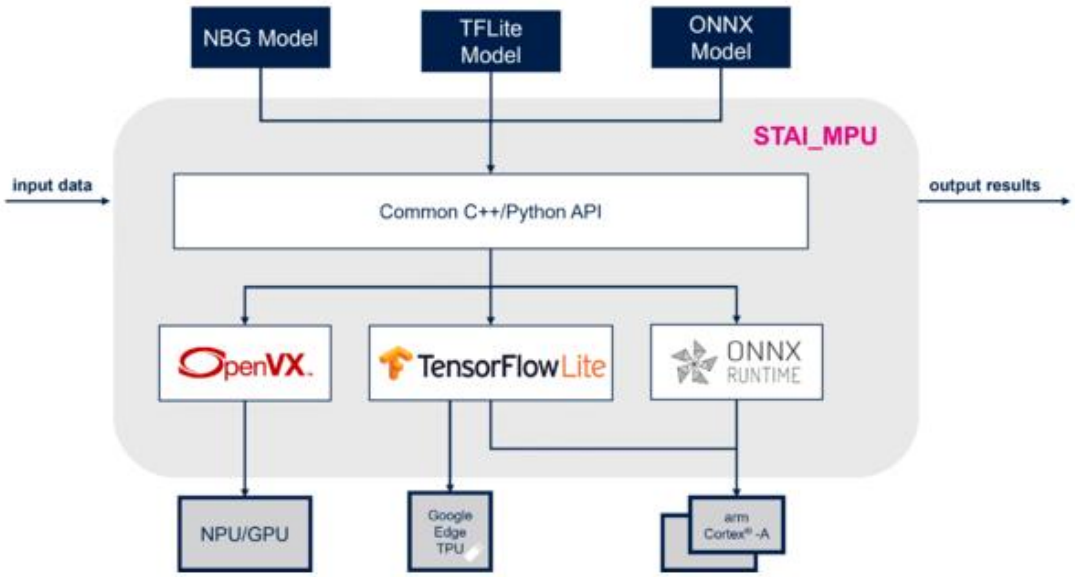
		STAI_MPU API				
AI framework		OpenVX		TensorFlowLite		ONNX RUNTIME
AI model type		Network Binary Graph (.nb)		TensorFlow Lite model (.tflite)		ONNX model (.onnx)
Execution engine		NPU	GPU	CPU	Coral EdgeTPU	CPU
Series	STM32MP25x	✓	✓	✓	✓	✓
	STM32MP15x	NA	NA	✓	✓	✓
	STM32MP13x	NA	NA	✓	✓	✓

常见 AI 框架的无缝集成：



X-LINUX-AI 支持模型有 ONNX 、Tensorflow Lite 、OpenVX 。

ST AI MPU 框架结构如下：



10.1.2 Yocto 下编译 X-LINUX-AI

```
~/stm32mp2_yocto_v24.11.06$ cd layers/meta-st

stm32mp2_yocto_v24.11.06/layers/meta-st$ git clone
```

```
https://github.com/STMicroelectronics/meta-st-x-linux-ai.git -b v6.0.1
stm32mp2_yocto_v24.11.06/layers/meta-st$ cd ../..
~/stm32mp2_yocto_v24.11.06$  source layers/meta-st/scripts/envsetup.sh
~/stm32mp2_yocto_v24.11.06/build-openstlinuxweston-stm32mp2$  bitbake-layers
add-layer ../layers/meta-st/meta-st-x-linux-ai/
build-openstlinuxweston-stm32mp2$  bitbake st-image-ai-npu
```

编译完成后生成

st-image-ai-npu-openstlinux-weston-stm32mp2.rootfs.ext4

st-image-ai-npu-userfs-openstlinux-weston-stm32mp2.userfs.ext4

st-image-ai-npu-vendorfs-openstlinux-weston-stm32mp2.vendorfs.ext4

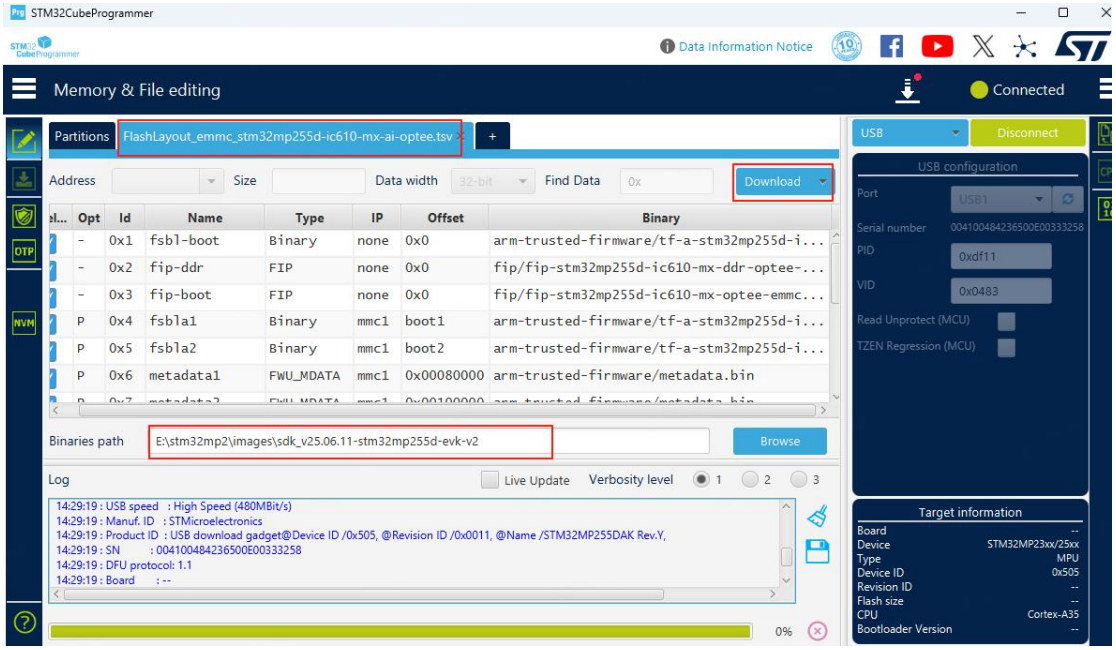
10.2 IC610 NPU

集成 VeriSilicon GC8000UL - TensorFlowLite - ONNX - Linux NN，900 MHz 时可达 1.35 TOPS 算力。

10.3 AI Image 烧录

Ai 镜像烧录包为 sdk_v24.11.06-stm32mp255d-evk-v2

参考 1.2 《STM32CubeProgrammer 烧录镜像》进入烧录模式，烧录 tsv 文件选择 FlashLayout_emmc_stm32mp255d-ic610-mx-ai-optee.tsv



10.4 AI demo 测试

10.4.1 图像分类 image classification

核心问题： “这张图片是什么？”

任务描述： 给定一张输入图像，算法需要判断它属于预定义类别中的哪一个（例如，猫、狗、汽车、飞机等）。这是最基础也是最经典的计算机视觉任务。

输出结果： 一个单一的标签（Label）和一个置信度（Confidence Score）。

例子： 输入一张猫的图片，模型输出“猫”（置信度 99%）。

经典模型： AlexNet, VGG, ResNet, EfficientNet 等。

当前模型已经支持的类别查询：

/usr/local/x-linux-ai/image-classification/models/mobilene/labels_imagenet_20

12.txt

摄像头输入，镜头对准电风扇自动识别：

```
/usr/local/x-linux-ai/image-classification/launch_bin_image_classification.sh
```

Ctrl+c 退出程序，accuracy 为置信度数值。

识别本地目录/usr/local/x-linux-ai/image-classification/models/mobilenet/testdata/内图片：

```
/usr/local/x-linux-ai/image-classification/launch_bin_image_classification_testdata.sh
```

10.4.2 物体检测 Object detection

核心问题： “图片里有什么？它们在哪里？”

任务描述： 不仅要识别出图片中的物体是什么（分类），还要用边界框（Bounding Box）精确地定位出每个物体的位置。一张图片中通常有多个物体。

输出结果： 多个边界框（每个框用坐标表示），以及每个框所对应物体的类别和置信度。

例子： 输入一张街景图，模型输出多个框，分别标出“汽车”、“行人”、“交通灯”等。

经典模型： R-CNN 系列, YOLO, SSD, RetinaNet 等。

当前模型已经支持的类别查询：

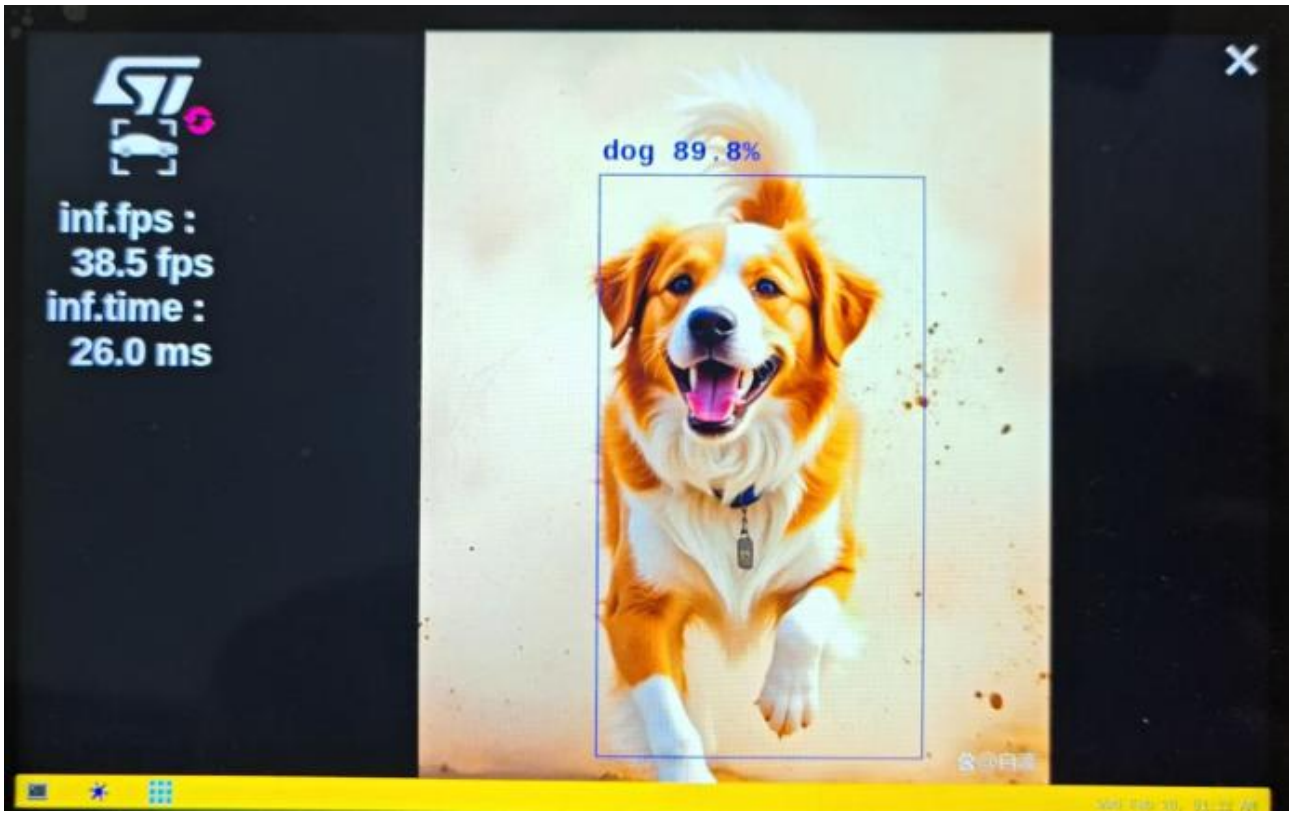
/usr/local/x-linux-ai/object-detection/models/coco_ssd_mobilenet/labels_coco_dataset_80.txt

摄像头采集输入物体检测：

```
/usr/local/x-linux-ai/object-detection/launch_bin_object_detection.sh
```

目 录 /usr/local/x-linux-ai/object-detection/models/coco_ssd_mobilenet/testdata ：

```
/usr/local/x-linux-ai/object-detection/launch_bin_object_detection_testdata.sh
```



10.4.3 姿态估计 pose estimation

核心问题： “图中人的关键关节在哪里？他们的姿势是怎样的？

任务描述： 识别图像或视频中人体（或动物、物体）的关键点（Keypoints），如肘部、

膝盖、手腕等，并连接这些点来构建骨骼模型，从而理解主体的姿态和动作。

输出结果： 一组关键点的坐标（x, y）及其连接关系。

应用： 动作识别、人机交互、体育分析、动画制作等。

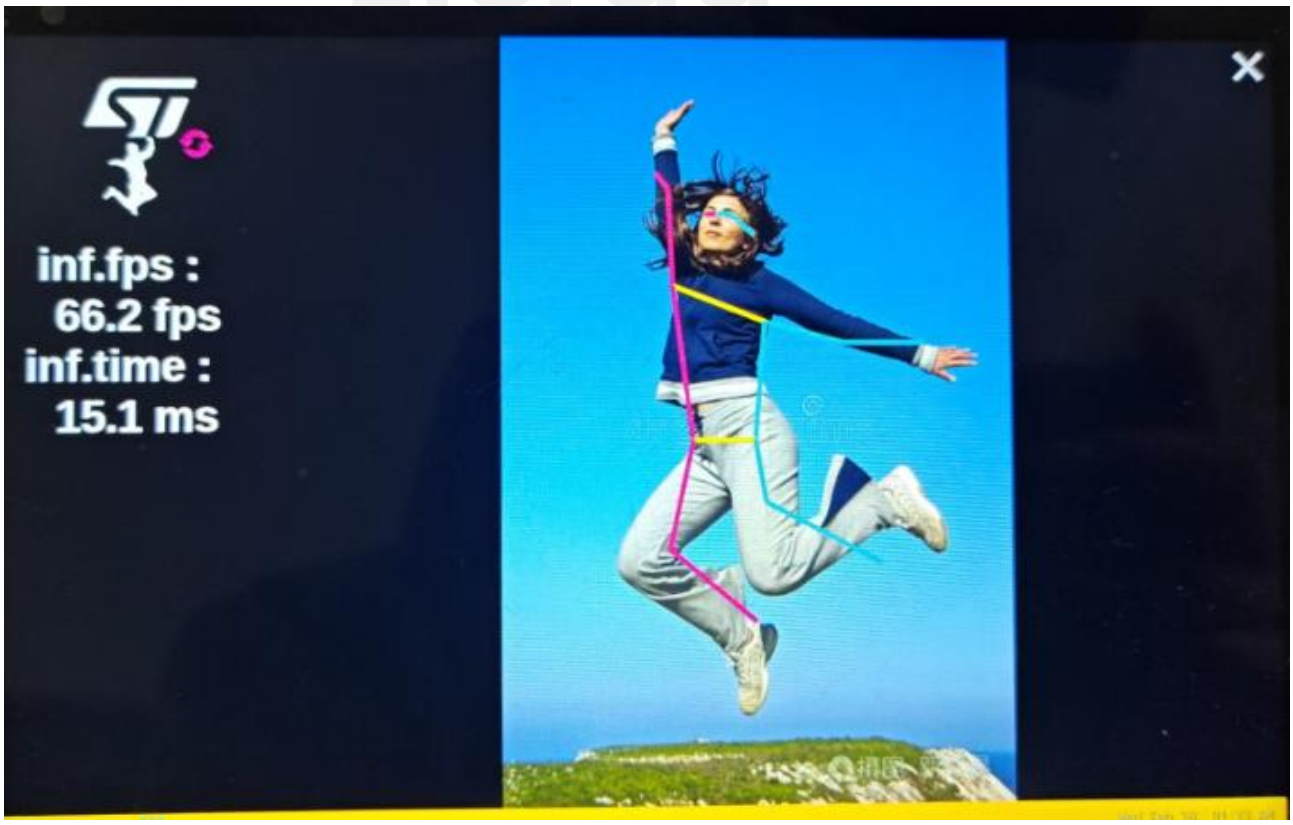
经典方法： OpenPose, HRNet, MMPose 等。

摄像头输入：

```
/usr/local/x-linux-ai/pose-estimation/launch_python_pose_estimation.sh
```

目录/usr/local/x-linux-ai/pose-estimation/models/yolov8n_pose/testdata：

```
/usr/local/x-linux-ai/pose-estimation/launch_python_pose_estimation_testdata.sh
```



10.4.4 语义分割 Semantic segmentation

核心问题： “图片中的每个像素属于哪个类别？”

任务描述： 对图像进行像素级的分类，即为图像中的每一个像素打上一个类别标签。它不区分同一类别的不同实例，只关心像素的类别。

输出结果： 一张与输入图片尺寸相同的分割掩膜（Segmentation Mask），其中每个像素的颜色代表其类别。

例子： 在自动驾驶中，将道路、天空、树木、行人、车辆等每一个像素都区分开来。

经典模型： FCN, U-Net, DeepLab 系列等。

当前模型已经支持的类别查询：

/usr/local/x-linux-ai/semantic-segmentation/models/deeplabv3/labels_pascalvoc.txt

图片放在

/usr/local/x-linux-ai/semantic-segmentation/models/deeplabv3/testdata

/usr/local/x-linux-ai/semantic-segmentation/launch_python_semantic_segmentation_testdata.sh

因版权问题，暂未放置图片，具体查看显示屏效果。

10.4.5 人脸识别 face recognition

核心问题： “这张脸是谁？” 或者 “这两张脸是同一个人吗？”

任务描述： 这是一个专门且复杂的任务，通常包含多个步骤：

人脸检测（Face Detection）：在图片中找出所有人脸的位置（目标检测的子任务）。

人脸对齐（Face Alignment）：校正人脸的角度，使眼睛、嘴巴等关键点对齐。

特征提取（Feature Extraction）：使用深度网络将对齐后的人脸图像转换为一个高维的、具有区分度的特征向量（Feature Embedding）。

特征匹配/识别（Matching/Recognition）：比较两个特征向量的相似度，来判断是否是同一个人（1:1 验证）或在数据库中查找最匹配的身份（1:N 识别）。

应用： 手机解锁、门禁系统、照片整理、公共安全。

将人脸图片保存在

/usr/local/x-linux-ai/face-recognition/models/facenet/testdata/

运行：

```
/usr/local/x-linux-ai/face-recognition/launch_bin_face_recognition_testdata.sh
```

因版权问题，暂未放置图片，具体查看显示屏效果。

Ctrl+c 退出程序

最大配置人数数量为 200

/usr/local/x-linux-ai/face-recognition/database

摄像头

```
/usr/local/x-linux-ai/face-recognition/launch_bin_face_recognition.sh
```

AI WIKI 链接：

https://wiki.stmicroelectronics.cn/stm32mpu/wiki/Category:X-LINUX-AI_expansion_package


Lierda
利尔达
期待反馈