

TW-STM32MP157-EVM 评估板

用户手册

版本：V1.0.1

修订历史

版本	日期	原因	修订者
V1.00	2021/06/21	创建文档	
V1.01	2022/03/07		

目 录

1. 产品介绍	4
1.1 产品简介	4
1.2 硬件资源介绍:	4
1.3 软件资源介绍	5
1.4 产品布局介绍	6
2. 开发板准备事项	8
2.1 上电前需要注意的事项	8
2.2 拨码开关设置及登录开发板	8
2.3 安装 SecureCRT 软件	8
2.3.1 安装 SecureCRT	8
2.3.2 使用 SecureCRT 登录	8
3. ST 官方软件安装	11
3.1 Java 环境安装	11
3.2 STM32CubeProgrammer 下载	12
3.2.1 STM32CubeProgrammer 的简介	12
3.2.2 STM32CubeProgrammer 软件下载	13
3.3 STM32CubeIDE 下载	18
3.3.1 STM32CubeIDE 的简介	18
3.3.2 STM32CubeIDE 软件下载	18
4. 烧写和更新固件	23
4.1 通过 STM32CubeProgrammer 烧写固件	23
4.2 通过 SD 卡进行烧写固件	25
5. 功能测试	28
5.1 LED 测试	28
5.2 蜂鸣器测试	28
5.3 串口测试	28
5.4 WIFI 测试	31
5.5 4G 测试	32
5.6 时钟设置	34
5.6.1 查看系统时钟:	34
5.6.2 查看 RTC 时钟:	34
5.6.3 设置 RTC 时钟:	34
5.6.4 同步系统时钟:	35
5.7 CAN 测试	35
5.8 摄像头测试	35
5.9 网络测试	36
5.10 音频测试	38
5.11 TF 卡测试	38
5.12 U 盘使用	39
5.13 USB 鼠标与 USB 键盘使用	40
5.14 LCD 背景亮度调节	41

5.15 USB 接口测试	41
5.16 触摸屏测试	42
5.17 CPU 温度	42
5.18 CPU 主频	42
5.19 查看 CPU 信息	43
5.20 查看内存信息	44
6. 环境搭建	45
6.1 安装虚拟机软件 VMware	45
6.2 交叉工具链	47
6.3 编译 Helloworld 源程序	48
6.4 编译 Uboot	49
6.5 编译内核	50
6.5.1 内核源码简介	50
6.5.2 内核配置	51
6.5.3 内核编译	54
6.6 搭建 QT 编译环境	55
6.7 编译 QT 的程序	60
7. 免责声明	64

1. 产品介绍

1.1 产品简介



图 1.1 TW-STM32MP157-EVM 评估板外观

1.2 硬件资源介绍：

表 1.1 TW-STM32MP157 评估板参数表

产品名称	TW-STM32MP157 评估板
操作系统	Linux
处理器	STM32MP157A
主频	A7 主频 650MHz, M4 主频 209MHz
内存	1GB
电子硬盘	2~8GB
3D-GPU	支持
WiFi	RTL8723DS
摄像头	1 路, CSI, 可扩展模拟摄像头
LCD 最高分辨率	1366 * 768

LVDS	一路 LCD 转 LVDS
触摸屏	电容触摸屏
音频	wm8960 立体声编解码器
USB	3 路 USB2.0
串口	最高 8 路（复用）
CAN-Bus	2 路
以太网	1 路
ADC	2 通道
DAC	2 通道
SD 卡接口	最高 2 路（复用）
I2C	6 路
SPI	6 路
SAI	4 路
GPIO	176
电源供电	直流+12V 电源供电

注：受限子评估底板的尺寸与接口布局，核心板部分资源以插针方式引出。

1.3 软件资源介绍

TW-STM32MP157-EVM 评估板提供完善的 Linux BSP，包括 Linux 内核源码、和开发工具等，具体软件资源如下表所示：

表 1.2 TW-STM32MP157-EVM 评估板软件资源表

软件资源		说明
Linux 内核		5.4.31
文件系统		根文件系统采用 ext4，在根文件系统上可挂载多种文件系统，如：sysfs、yaffs2、ubifs 等
交叉编译器（内核）		arm-none-linux-gnueabi-gcc 9.2.1
交叉编译器（应用程序）		arm-none-linux-gnueabi-gcc 9.2.1
外设驱动	eMMC	驱动源码：/drivers/mmc
	SD/MMC	驱动源码：/drivers/mmc
	LCD	驱动源码：/drivers/gpu/drm/panel/panel-simple.c
	触摸屏	驱动源码：/drivers/input/touchscreen/
	摄像头	驱动源码：/drivers/media/platform/mxc/capture/

	I2C	驱动源码: i2c/busses/i2c-stm32f7.c
	UART	驱动源码: drivers/tty/serial/stm32-usart.c
	USB	驱动源码: /drivers/usb/host/xxx-platform.c
	以太网	驱动源码: drivers/net/ethernet/stmicro/stmmac/dwmac-stm32.c
	CAN	驱动源码: drivers/net/can/m_can/m_can.c
	WIFI	驱动源码: /drivers/net/wireless
	PWM	驱动源码: /drivers/pwm
	GPIO	驱动源码: /drivers/gpio
	RTC	驱动源码: /drivers/rtc
图形界面	使用 QT5.14.2	
工具软件	如系统镜像烧写工具、串口调试工具、网络调试工具、tftp 服务器软件等	

1.4 产品布局介绍

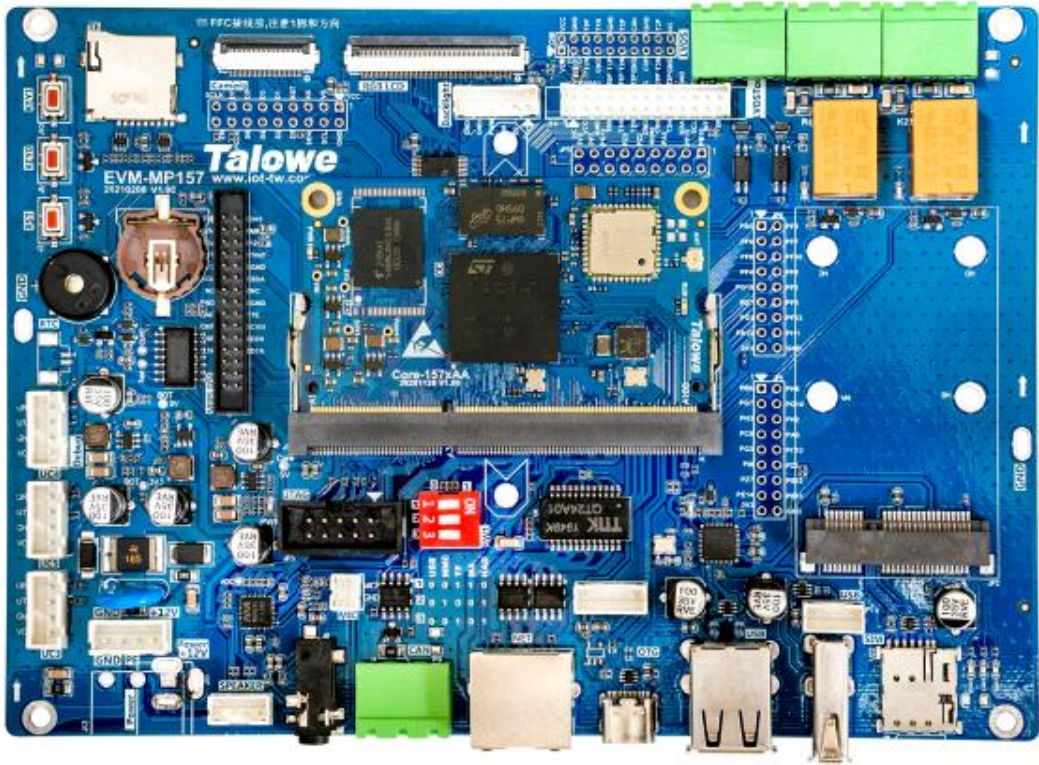


图 1.2 TW-STM32MP157-EVM 评估板布局图

注:图片仅供参考,以实际销售产品为准

表 1.4 评估底板跳线与指示灯说明

序号	评估底板标号	描述
----	--------	----

1	LED1	运行指示灯，GPIO 控制
2	PWR	电源指示灯
3	RST	主板复位按键

2. 开发板准备事项

2.1 上电前需要注意的事项

1. 首先检查电源是否插上，如果电源不插上，仅由 USB_TTL 供电可能会导致整个开发板供电不足，导致 LCD 或者 HDMI 无法显示和 LCD 无法触摸及 4G 模块无法正常工作等情况。
2. 防止底板接触到金属异物，将底板下的某两个触点短路。在使用久了的情况下也要注意观察有没有异物落在开发板的上面，以防短路，注意防水，防潮，防尘等

2.2 拨码开关设置及登录开发板

序号	U15 拨码顺序			描述
	3	2	1	
1	off	on	off	EMMC 启动
2	off	off	off	USB 启动
3	on	off	on	SD/TF 启动
4	off	off	on	Cortex-M4

表 2.1 拨码开关启动选择说明

硬件连接如下：

- (1) 断开 TW-STM32MP157-EVM 评估板的供电电源；
- (2) 把 TW-STM32MP157-EVM 评估板设置为 EMMC 启动方式(U15 拨码开关拨到 off on off)；

2.3 安装 SecureCRT 软件

2.3.1 安装 SecureCRT

用户可以从“TW-STM32MP157 光盘资料\2.软件开发参考资料\5.工具软件\1.SecureCRT\SecureCRT”下载 SecureCRT 安装程序。直接将 SecureCRT 文件夹放到电脑的任意位置，点击文件夹中的 SecureCRT.exe 执行程序即可使用。

2.3.2 使用 SecureCRT 登录

运行 SecureCRT 软件，弹出“连接”对话框，如下图 2-1 所示：

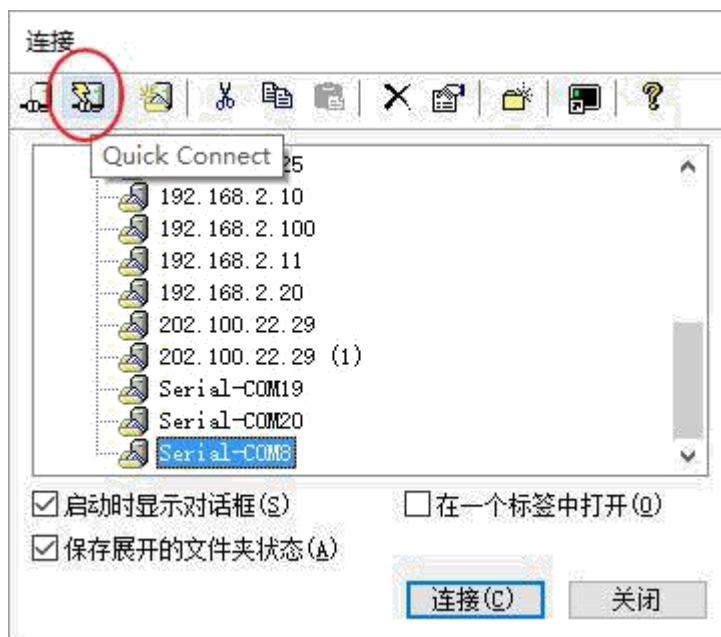


图 2.1 SecureCRT “连接”对话框

点击“Quick Connect”按钮，弹出“快速连接”对话框，如下图所示：



图 2.2 SecureCRT “快速连接”对话框

在该对话框中，选择“协议”为“Serial”，“端口”需根据主机实际使用的串口号进行选择（可从操作系统的“设备管理器”中获得该信息），并对串口参数进行相应的设置，值得注意的是在设置串口参数时需关闭所有的“数据流控制”选项。在设置完成后，点击“连接”按钮，会出现一个用于登录的 SecureCRT 终端窗口。

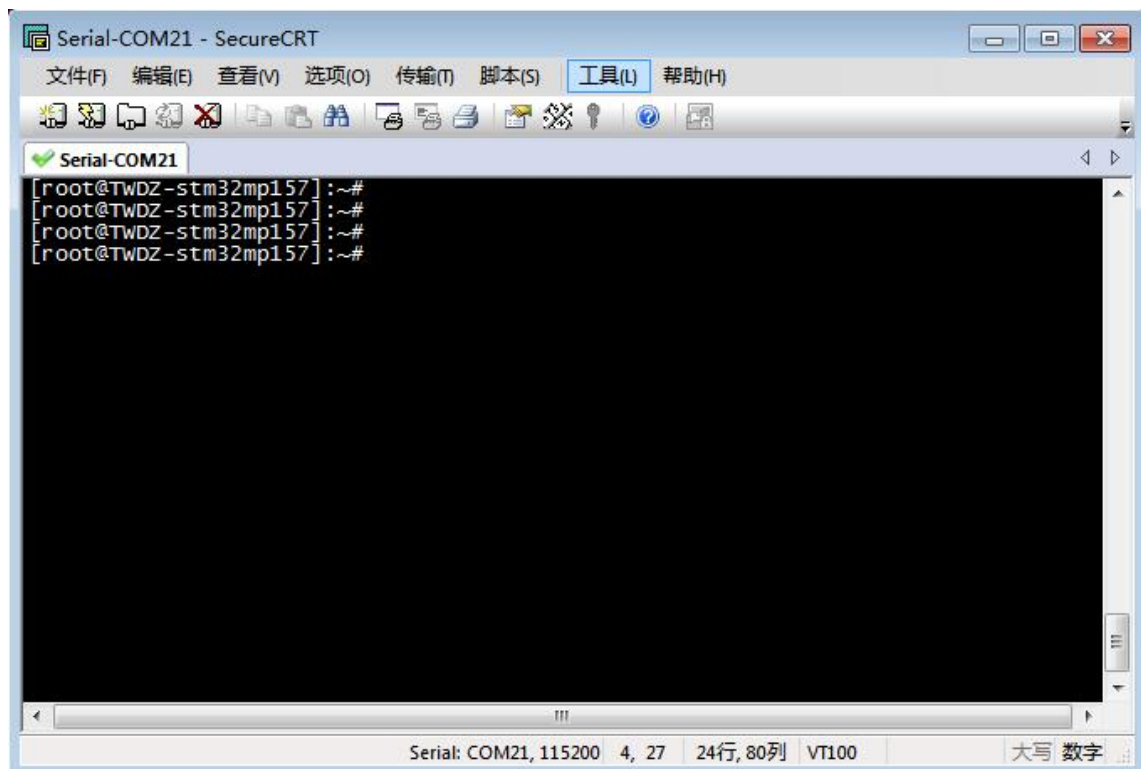


图 2.3 SecureCRT 终端登录窗口

3. ST 官方软件安装

3.1 Java 环境安装

在 Window 安装 STM32CubeProgrammer 软件需要安装 Java 环境才能正常使用。如果没有安装 Java 软件包，在打开的 STM32CubeProgrammer 软件的时候会报以下错误。



图 3.1 STM32CubeProgrammer 软件打开错误

可以在路径为“TW-STM32MP157 光盘资料\2. 软件开发参考资料\5. 工具软件\固件烧录软件”中选中 jre-8u291-windows-x64.exe 进行安装。

更改目标文件夹后点击安装，如果使用默认安装路径直接点击安装即可



图 3.2 Java 环境的安装



图 3.3 Java 安装过程

安装完成后，直接点击关闭



图 3.4 Java 安装完成

安装完 Java 环境后，STM32Cube 系列软件可以正常打开使用。

3.2 STM32CubeProgrammer 下载

3.2.1 STM32CubeProgrammer 的简介

STM32CubeProgrammer 简称 STM32CubeProg，是一个适用于 STM32 系列产品的跨平台、多合一的程序烧写工具。有如下特点：

(1) “跨平台” 体现在支持 Windows、macOS 和 Linux 操作系统，软件运行时需要 Java 环境。

(2) “多合一” 体现在支持通过 USB、ST-LINK、UART、OTA 多种方式来烧写固件。

3.2.2 STM32CubeProgrammer 软件下载

我们需要在 Windows 使用上位机通过 USB Type-C 连接线连接到 TW-STM32MP157 开发板，然后进行烧写固件操作，所以我们就需要这样的上位机软件，STM32CubeProgrammer 是由 ST 官方提供的，适用于烧写 ST 各种平台，现在我们只讲用 USB Type-C 连接方法烧写烧写固件到 TW-STM32MP157 开发板的方法。路径为“TW-STM32MP157 光盘资料\2.软件开发参考资料\5.工具软件\固件烧录软件”。

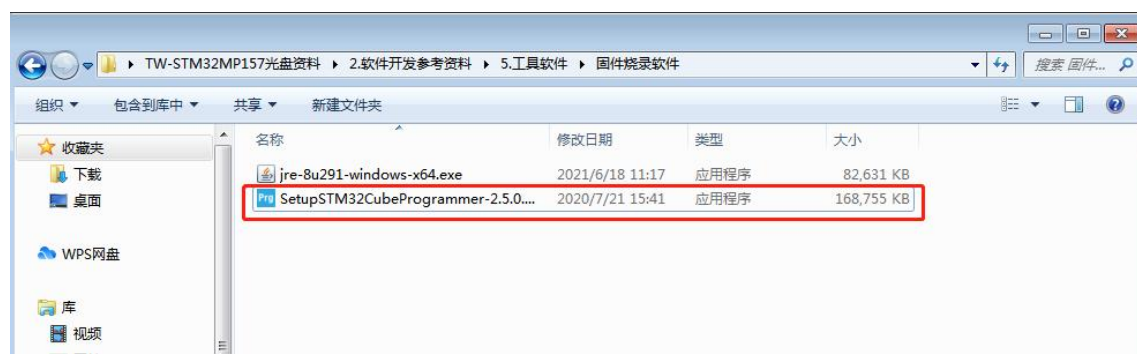


图 3.5 STM32CubeProgrammer 软件

点击 next

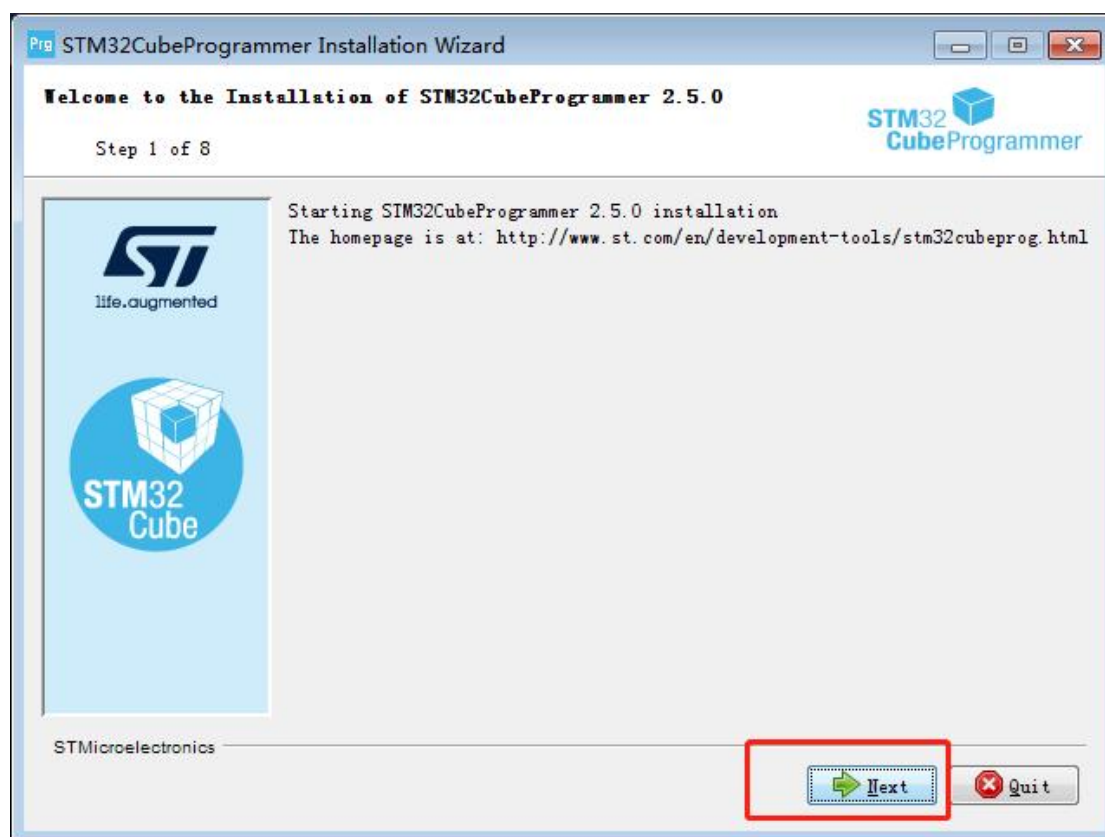


图 3.6 STM32CubeProgrammer 欢迎界面



图 3.7 STM32CubeProgrammer 安装信息

选择第 1 选项，然后 next

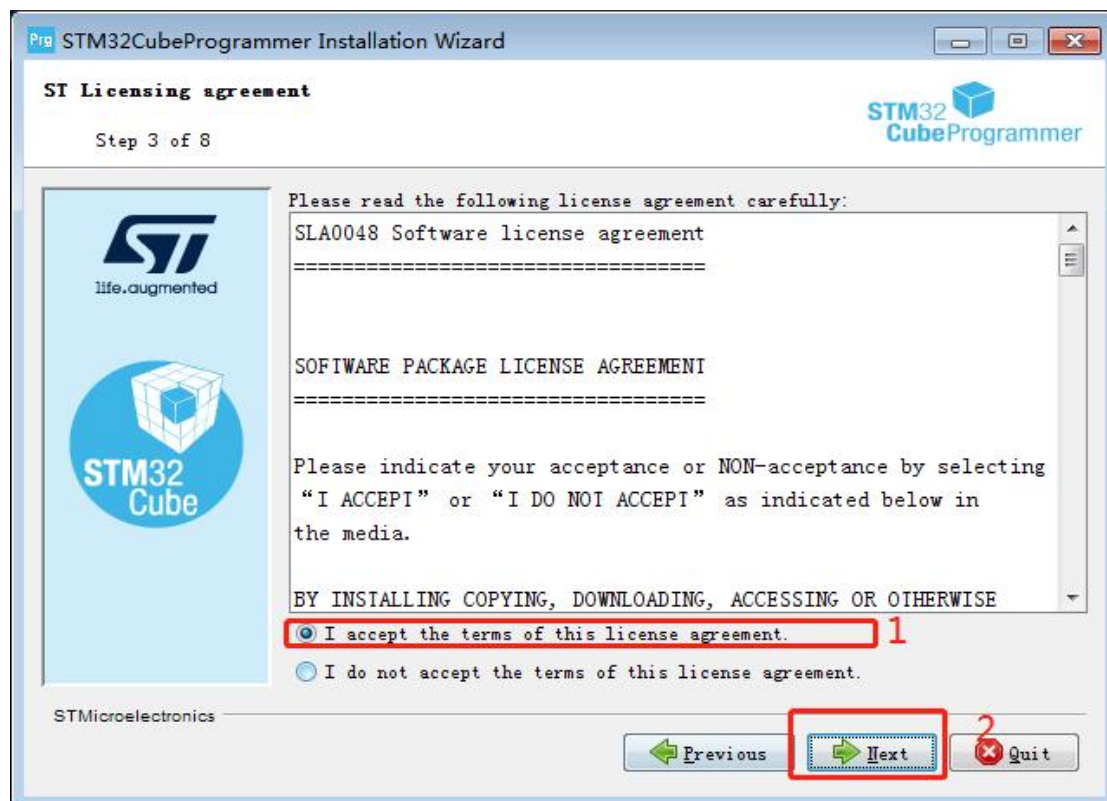


图 3.8 STM32CubeProgrammer 软件条款同意

选择软件的安装路径，然后 next

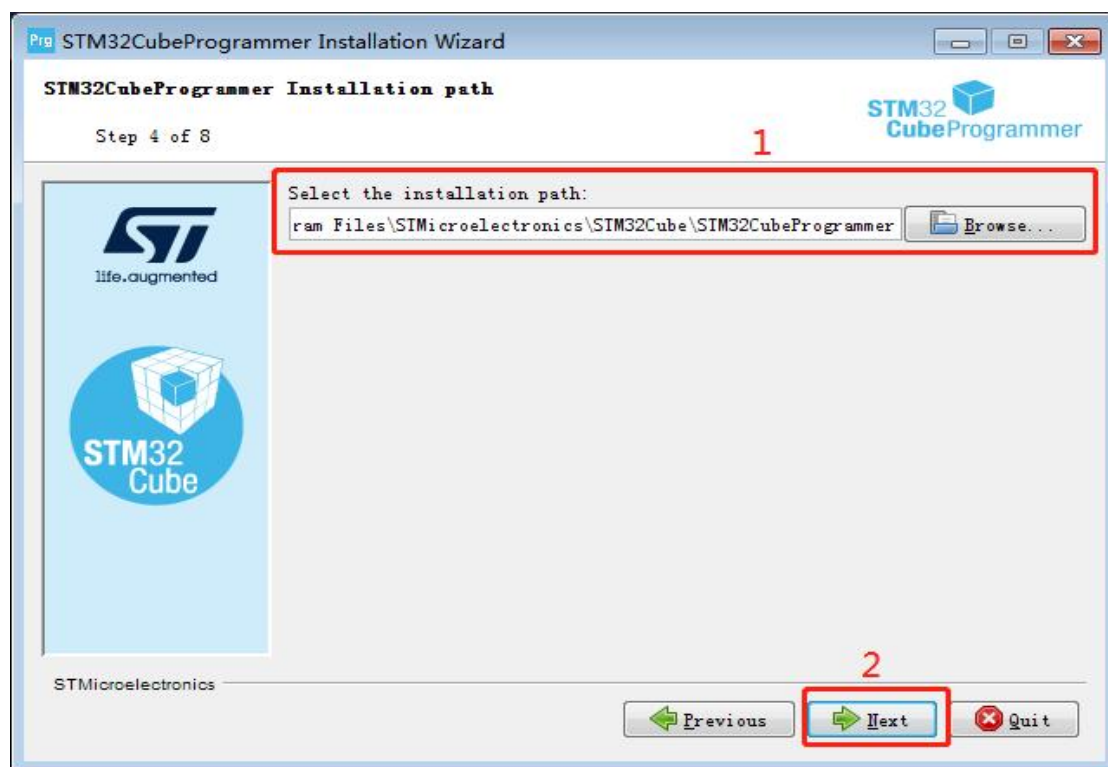


图 3.9 STM32CubeProgrammer 安装路径

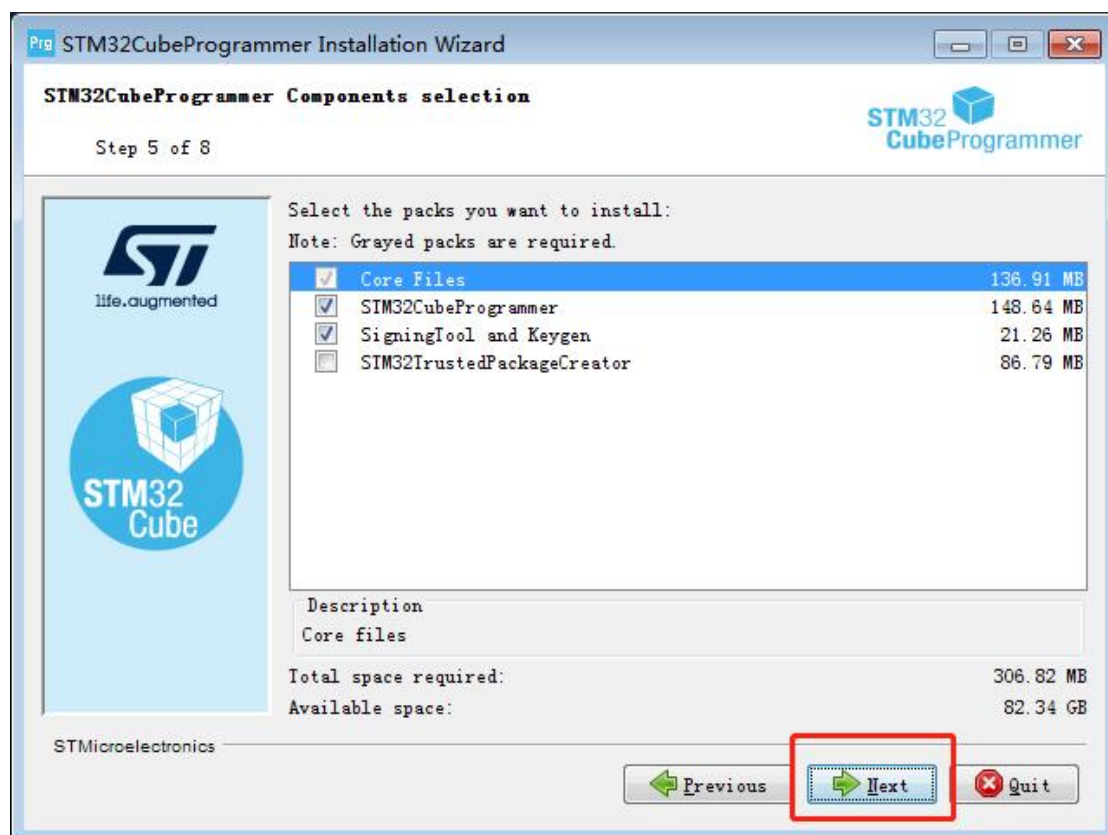


图 3.10 STM32CubeProgrammer 默认安装选项

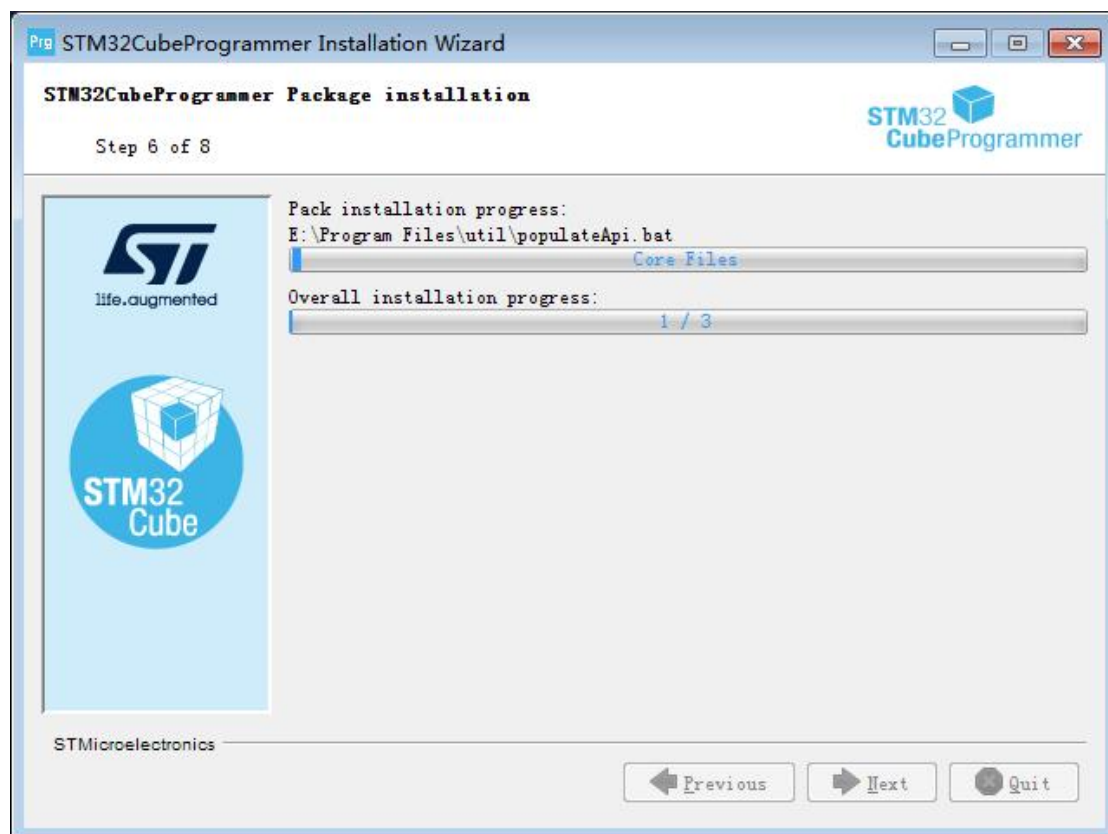


图 3.11 STM32CubeProgrammer 安装过程

在安装的时候，需要安装相关的驱动软件，选择“始终安装此驱动程序软件”



图 3.12 驱动软件的安装

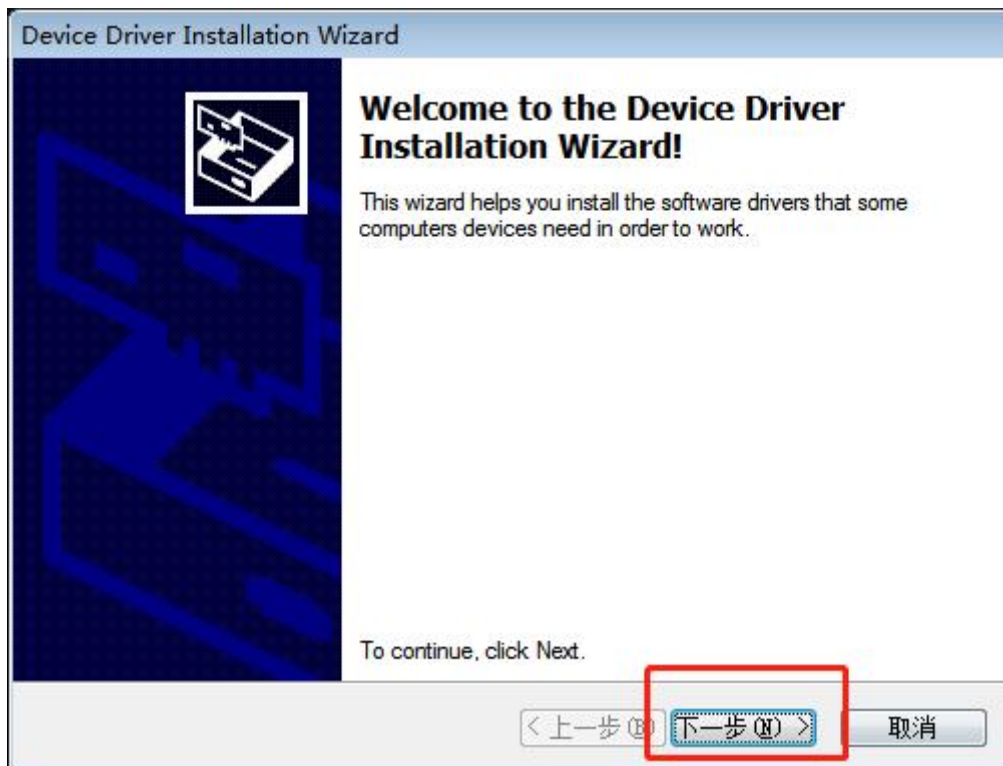


图 3.13 驱动安装界面

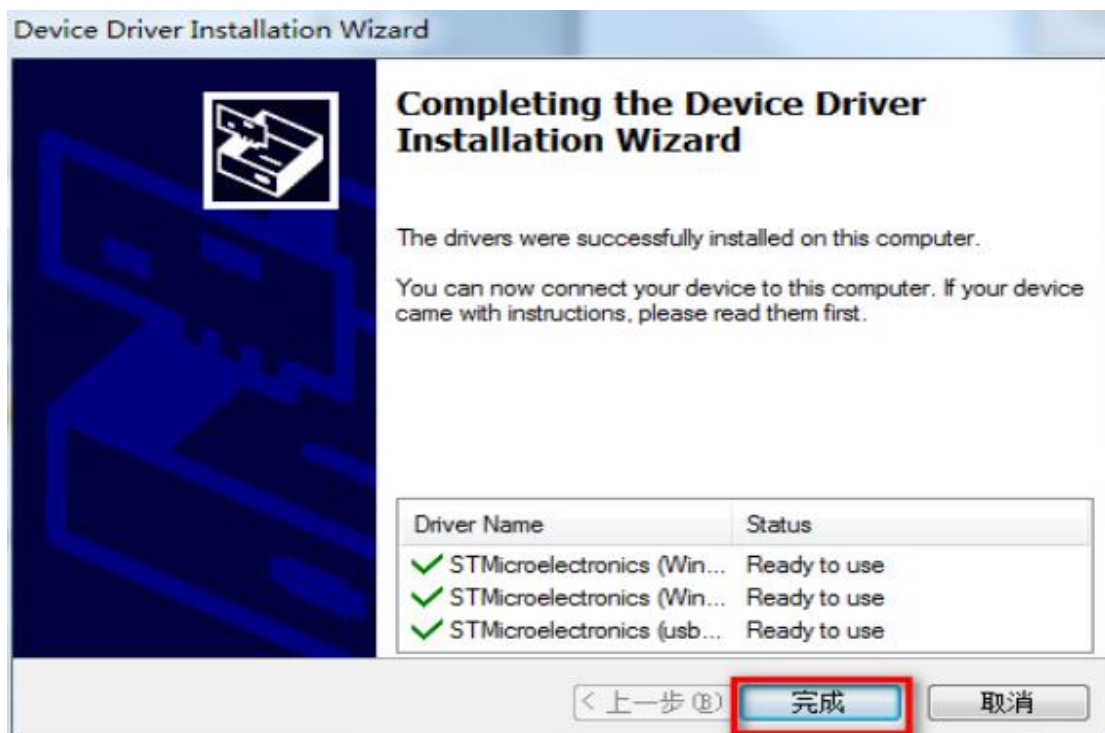


图 3.14 驱动安装完成

3.3 STM32CubeIDE 下载

3.3.1 STM32CubeIDE 的简介

1. STM32CubeIDE 是一个先进的 C/C++ 开发平台，具有 STM32 微控制器的 IP 配置，代码生成，代码编译和调试功能。

2. 它基于 ECLIPSE™/ CDT 框架和用于开发的 GCC 工具链，以及用于调试的 GDB。它允许集成数百个现有插件，完成 ECLIPSE™IDE 的功能。

3.3.2 STM32CubeIDE 软件下载

光盘资料提供了 STM32CubeIDE 软件的安装包,路径为“TW-STM32MP157 光盘资料\2. 软件开发参考资料 \5. 工具软件 \固件烧录软件 ” 的 st-stm32cubeide_1.4.0_7511_20200720_0928_x86_64.exe。STM32CubeIDE 的安装比较简单，基本一路 Next 下去就行，该勾选的勾选上即可。注意:(STM32CubeIDE 安装包不能放在带中文路径下，否则会报以下图 3.15 的错误)



图 3.15 安装包有中文路径错误

点击 next



图 3.16 STM32CubeIDE 欢迎界面

点击 I Agree

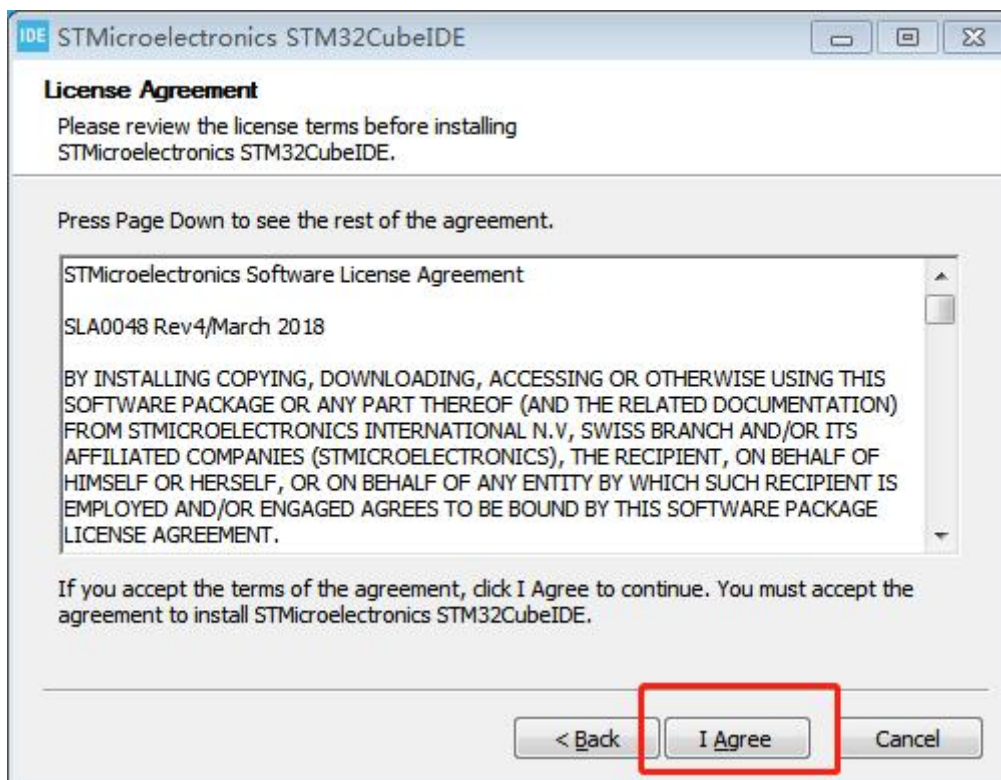


图 3.17 STM32CubeIDE 安装协议

选择安装路径后，点击 next

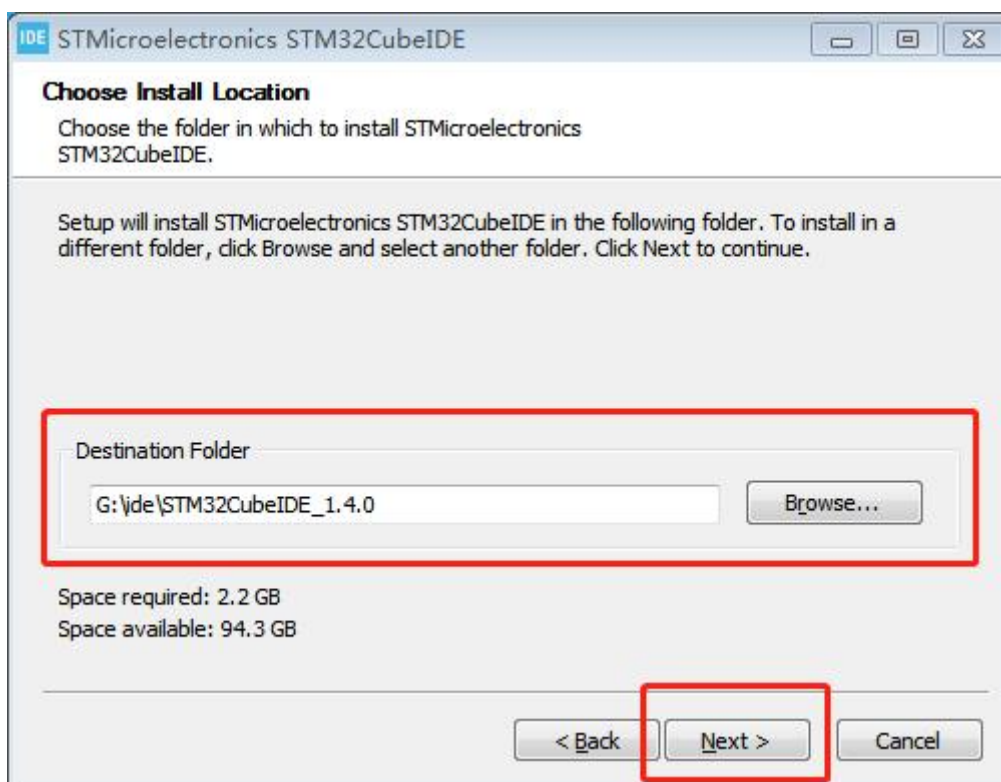


图 3.18 STM32CubeIDE 安装路径

点击 install

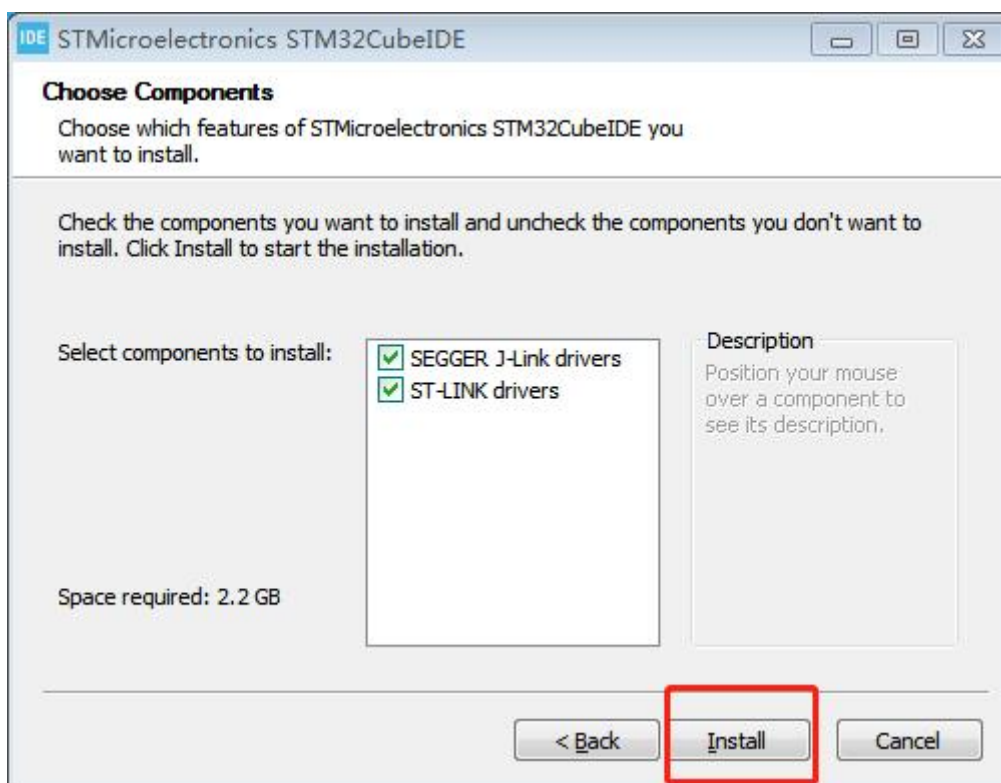


图 3.19 STM32CubeIDE 安装配置选项

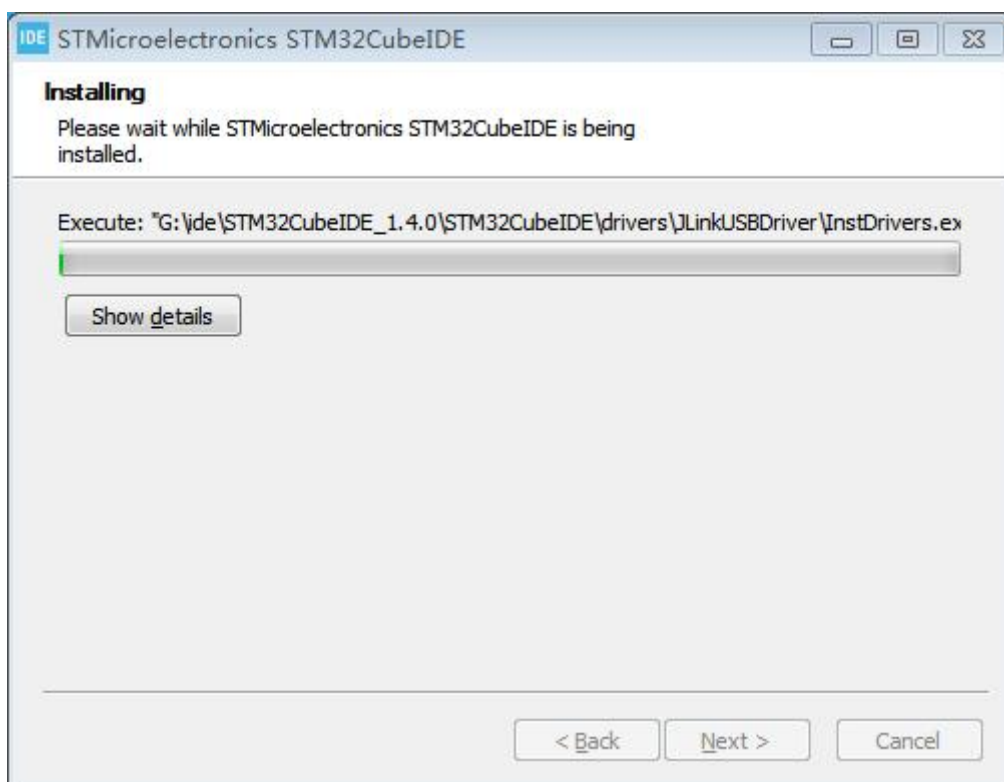


图 3.20 STM32CubeIDE 正在安装中

点击 next

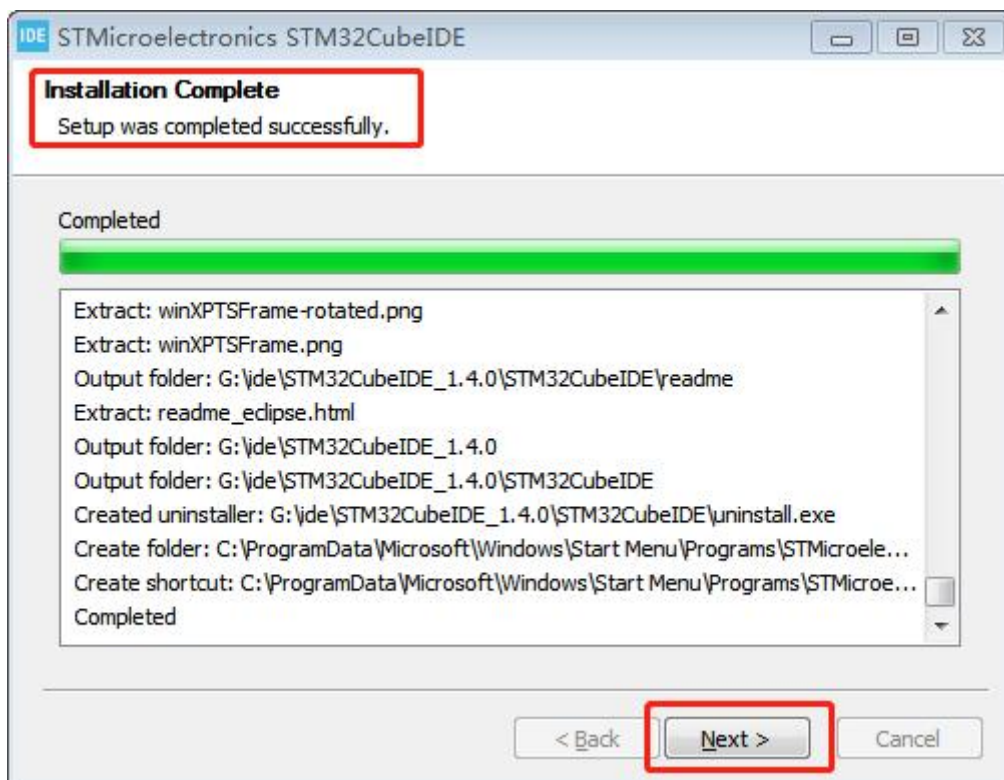


图 3.21 STM32CubeIDE 安装完成



图 3.22 STM32CubeIDE 创建桌面快捷方式

4. 烧写和更新固件

4.1 通过 STM32CubeProgrammer 烧写固件

注意:在此之前,需要完成 [3.ST 官方软件安装](#) 章节才能进行烧写固件。

1. 双击打开 STM32CubeProgrammer 软件

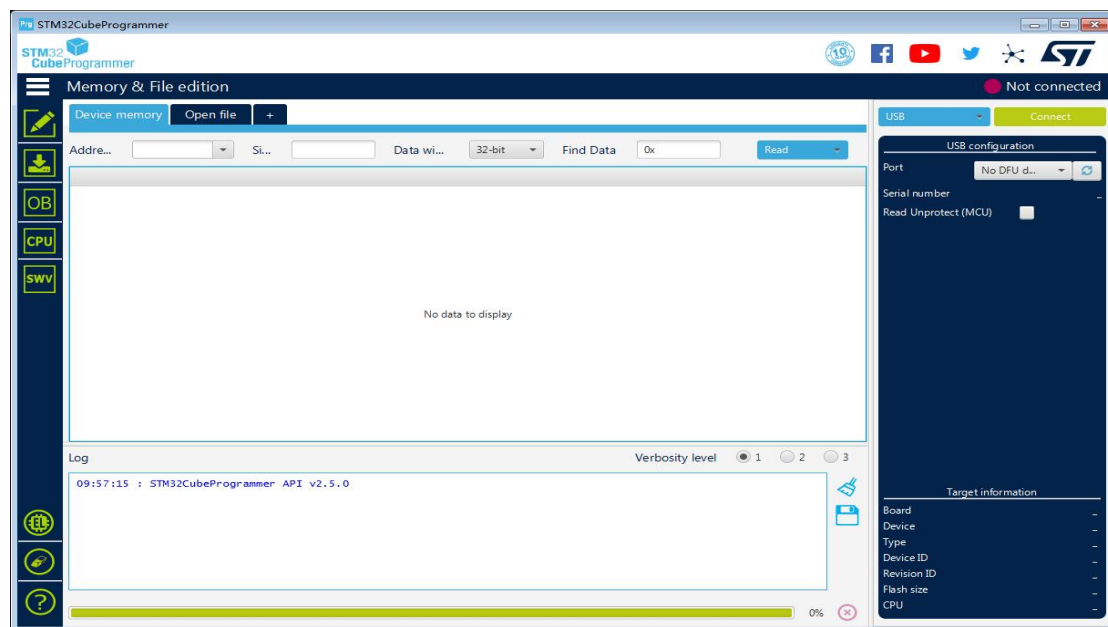


图 4.1 STM32CubeProgrammer 软件界面

2. 将启动模式设为 USB 模式, 查看(2.2 拨码开关设置及登录开发板), 然后点击 Connect 键

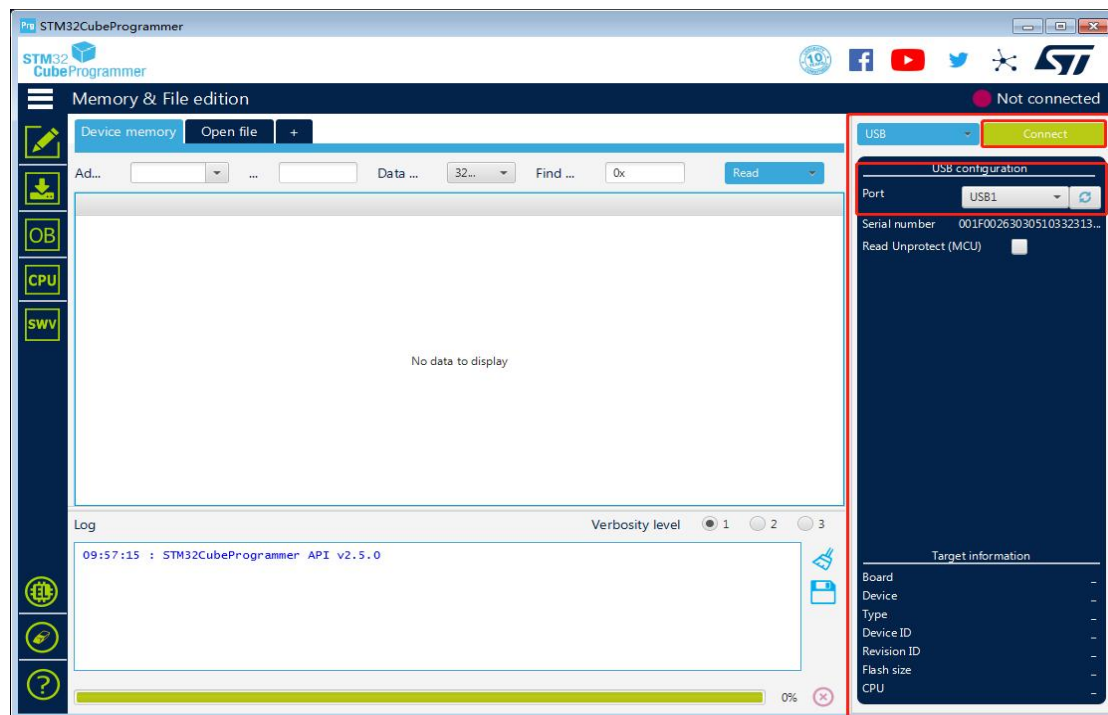


图 4.2 STM32CubeProgrammer 连接 USB

3. 选中下载的固件 emmc_stm32mp157_tw.tsv 配置文件，路径为 “TW-STM32MP157 光盘资料\4.固件更新与烧写方法\1.USB 烧写\2.标准固件\MP1_IMAGE”

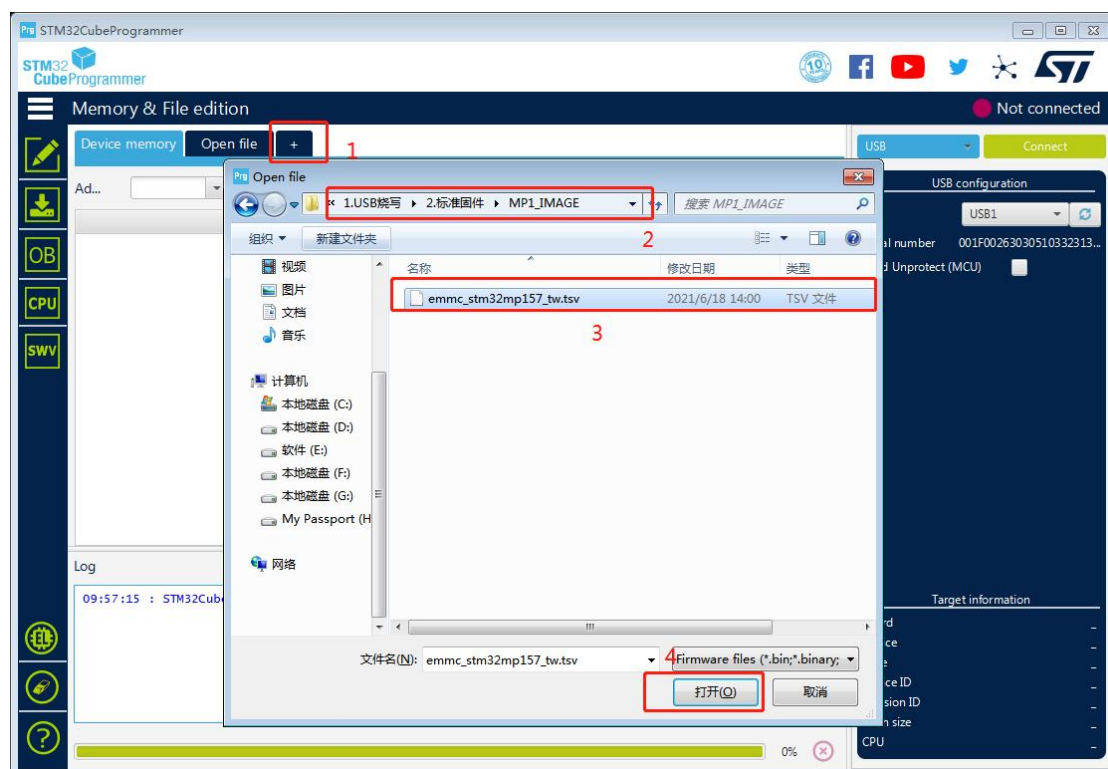


图 4.3 STM32CubeProgrammer 导入配置文件

4. Binaries path 的路径是下载的固件路径，选择正确后，按 Download 键进行下载。

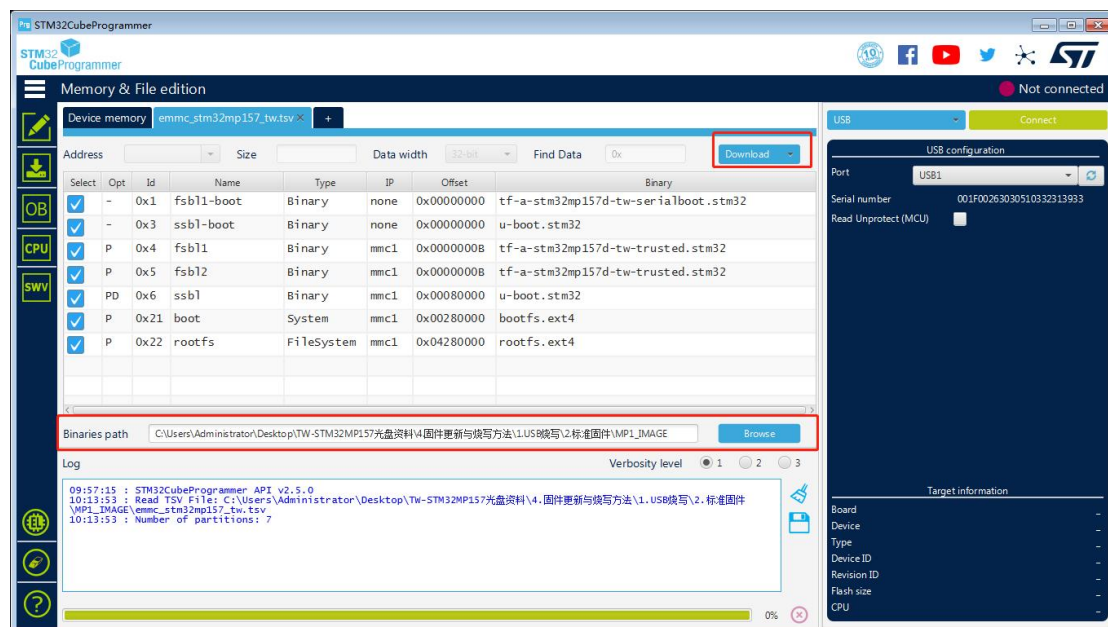


图 4.4 STM32CubeProgrammer 烧写固件

有时候在烧写固件的时候，并不想将所有的固件重新烧写一次，然后可以通过修改烧写脚本来进行部分烧写。

1. 打开烧写脚本 `emmc_stm32mp157_tw.tsv`, 路径为 “TW-STM32MP157 光盘资料\4.固件更新与烧写方法\1.USB 烧写\2.标准固件\MP1_IMAGE”

#	Opt	Id	Name	Type	Device	Offset	Binary
1	-	0x01	fsbl1-boot	Binary	none	0x0	tf-a-stm32mp157d-tw-serialboot.stm32
2	-	0x03	ssbl-boot	Binary	none	0x0	u-boot.stm32
3	P	0x04	fsbl1	Binary	mmc1	boot1	tf-a-stm32mp157d-tw-trusted.stm32
4	P	0x05	fsbl2	Binary	mmc1	boot2	tf-a-stm32mp157d-tw-trusted.stm32
5	PD	0x06	ssbl	Binary	mmc1	0x00080000	u-boot.stm32
6	P	0x21	boot	System	mmc1	0x00280000	bootfs.ext4
7	P	0x22	rootfs	FileSystem	mmc1	0x04280000	twdz_rootfs_emmc.ext4

图 4.5 emmc_stm32mp157_tw.tsv 内容

域	作用
Opt	选项字段，可以设置为“-”、“P”、“D”或“E”
Id	会根据这个 id 来决定烧写分区
Name	分区名字
Type	制定烧写的类型，仅 uboot 使用。
Device	指定烧写的设备类型与编号，比如 emmc0、emmc1、nand0 等，如果 opt 为 ‘-’ ,那么此字段就为 none
Offset	分区的起始位置，如果为 “boot1” 表示 EMMC 的第一个分区，如果为 “boot2” 就表示 EMMC 第二个分区。如果是数字就表示需要偏移的字节数。
Binary	要烧录的文件

表 4.1.1 emmc_stm32mp157_tw.tsv 的配置解析

2. 通过配置 Opt 项，可以选择烧写指定固件

‘-’：空选型。

‘P’：更新分区或设备，也就是向分区或设备烧写固件。

‘PE’：不更新，也就是指定某个分区或者设备不需要烧写固件，这样我们就可以单独只更新 tf-a、uboot、kernel 或者 rootfs。

‘PD’：删除并更新，也可以写作 DP。

‘PDE’：删除并且保持为空，也可以写作 PED/DPE/DEP/EPD/EDP。

4.2 通过 SD 卡进行烧写固件

通过 SD 卡烧写固件的实际速度要比 USB 烧写速度要快 3 分钟左右，所以，我们也可以通过 SD 卡去烧写或者更新 EMMC。

1. 将光盘资料路径为 “TW-STM32MP157 光盘资料\4.固件更新与烧写方法\2.TF 卡烧写\1.TF 卡制作\” 中的文件夹 SDupdate 拷贝到虚拟机上。

2. 准备一张 SD 卡烧写固件，存储大小至少 4GB（SD 卡需要格式化，所以在格式化前，将里面的文件备份好，以免丢失），/dev/sdb 是该 SD 卡的节点。

```
twdz@ubuntu:~$ ls /dev/sd*
/dev/sda /dev/sda1 /dev/sda2 /dev/sda5 /dev/sdb
twdz@ubuntu:~$
```

3. 执行 SDupdate 文件中的 sd_update.sh 脚本，执行指令：./sd_update.sh /dev/sdb。执行过程中会打印以下信息，看到“SD 启动卡制作完成”的字样即完成 SD 卡制作。

```
Disk identifier (GUID): 7C1CA90B-1281-4E1F-82D9-A6500EA5BE92
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 15554526
Partitions will be aligned on 1-sector boundaries
Total free space is 0 sectors (0 bytes)

Number  Start (sector)    End (sector)  Size      Code  Name
   1            34             545        256.0 KiB   8301  fsbl1
   2           546            1057        256.0 KiB   8301  fsbl2
   3          1058            5153         2.0 MiB    8301  ssbl
   4          5154          136225        64.0 MiB    8300  bootfs
   5         136226          341027       100.0 MiB    8300  rootfs
   6         341028         15554526       7.3 GiB    8300  emmc_rootfs

The operation has completed successfully.
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 65536 1k blocks and 16384 inodes
Filesystem UUID: fc37a255-57f6-455b-854a-b6e5bd5f50d5
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
twdz@ubuntu: ~/MP1/twdz源码包/SDupdate
usr/share/udhcpc/
usr/share/udhcpc/default.script
usr/share/dict/
usr/games/
usr/src/
var/
var/volatile/
var/run
var/spool/
var/spool/mail/
var/log
var/cache/
var/cache/ldconfig/
var/cache/ldconfig/aux-cache
var/cache/rpm/
var/lib/
var/lib/misc/
var/lib/urandom/
var/lock
var/backups/
var/local/
var/tmp
文件系统写入成功
SD启动卡制作完成.....
twdz@ubuntu:~/MP1/twdz源码包/SDupdate$
```

4. 将评估板拨码到 TF 卡启动模式，插入烧写过固件的 TF 卡，上电，当打印出以下信息并且伴随蜂鸣器响时，说明 TF 卡烧写完成。

```
Populating dev cache
Fri Aug 25 07:15:10 UTC 2017
INIT: Entering runlevel: 5
check file ok.....
write TF-A file
write TF-A file finish.....
write Uboot file
write u-boot file finish.....
write bootfs file
write bootfs file finish.....
write Rootfs file
write Rootfs file finish.....
Update complete...
Update complete...
```

5. 断电，将拨码拨回 EMMC 启动即可。

5. 功能测试

5.1 LED 测试

对开发板的 LED0、LED1 进行测试，串口终端执行相应的指令去控制对应的 IO 来控制对应的继电器和 LED 灯的亮灭，如下图所示。

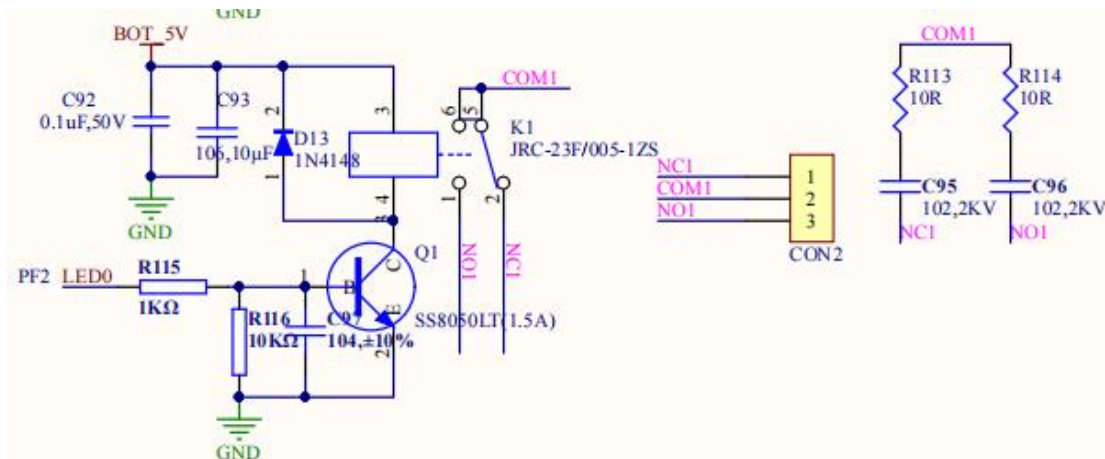


图 5.1 LED0 原理图

当执行以下指令后，开发板中 LED0 会亮起，COM1 和 NO1 连通

```
[root@TWDZ-stm32mp157]:~# echo 82 > /sys/class/gpio/export//申请 IO
[root@TWDZ-stm32mp157]:~# echo out > /sys/class/gpio/gpio82/direction//设置方向
[root@TWDZ-stm32mp157]:~# echo 1 > /sys/class/gpio/gpio82/value//设定输出值
```

同理可以测试 LED1 是否能正常使用。

5.2 蜂鸣器测试

通过指令对开发板的 Beep 进行测试，Beep 配置为普通 IO 口，通过控制 IO 口的高低电平可以驱动蜂鸣器，可在命令行下运行如下命令后，蜂鸣器会鸣叫：

```
[root@TWDZ-stm32mp157]:~# echo 1 > /sys/class/pwm/pwmchip8/export
[root@TWDZ-stm32mp157]:~# echo 1000000 > /sys/class/pwm/pwmchip8/pwm1/period
[root@TWDZ-stm32mp157]:~# echo 500000 > /sys/class/pwm/pwmchip8/pwm1/duty_cycle
[root@TWDZ-stm32mp157]:~# echo 1 > /sys/class/pwm/pwmchip8/pwm1/enable
```

5.3 串口测试

TW-STM32MP157-EVM 的底板上总共有三个串口。

串口名	对应串口	对应设备
UC4(调试串口)	Uart4	/dev/ttySTM0
UC3	Uart3	/dev/ttySTM1
UC5	Uart5	/dev/ttySTM2

1.通过下面的指令配置串口

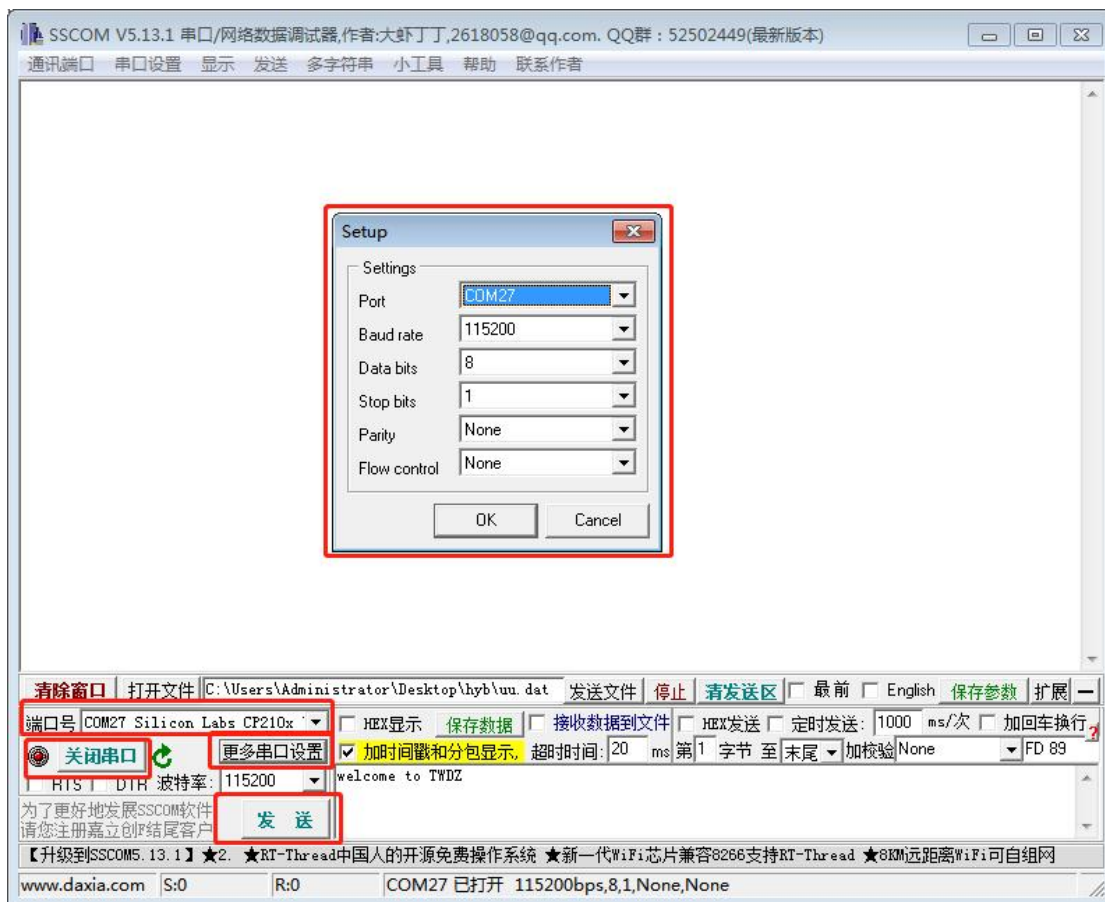
```
[root@TWDZ-stm32mp157]:~# stty -F /dev/ttySTM1 ispeed 115200 ospeed 115200 cs8
```

stty 指令解释:

- (1) -F(--file): 打开指定的设备,并用此设备作为输入来代替标准输入
- (2) ispeed N: 设置输入速率为 N
- (3) ospeed N: 设置输出速率为 N
- (4) csN: 把字符长度设为 N

2.在 PC 端打开串口调试工具,该工具可接收开发板串口发送的数据,并可发送数据到开发板。Windows 下的串口调试软件比较多,此处以 sscom32.exe 为例介绍串口调试工具的使用,该工具可在“TW-STM32MP157 光盘资料/2、软件开发参考资料/5、开发工具软件\SSCOM32”中直接获取。

打开 sscom32.exe 软件,并进行相应的配置,对话框如下图所示:



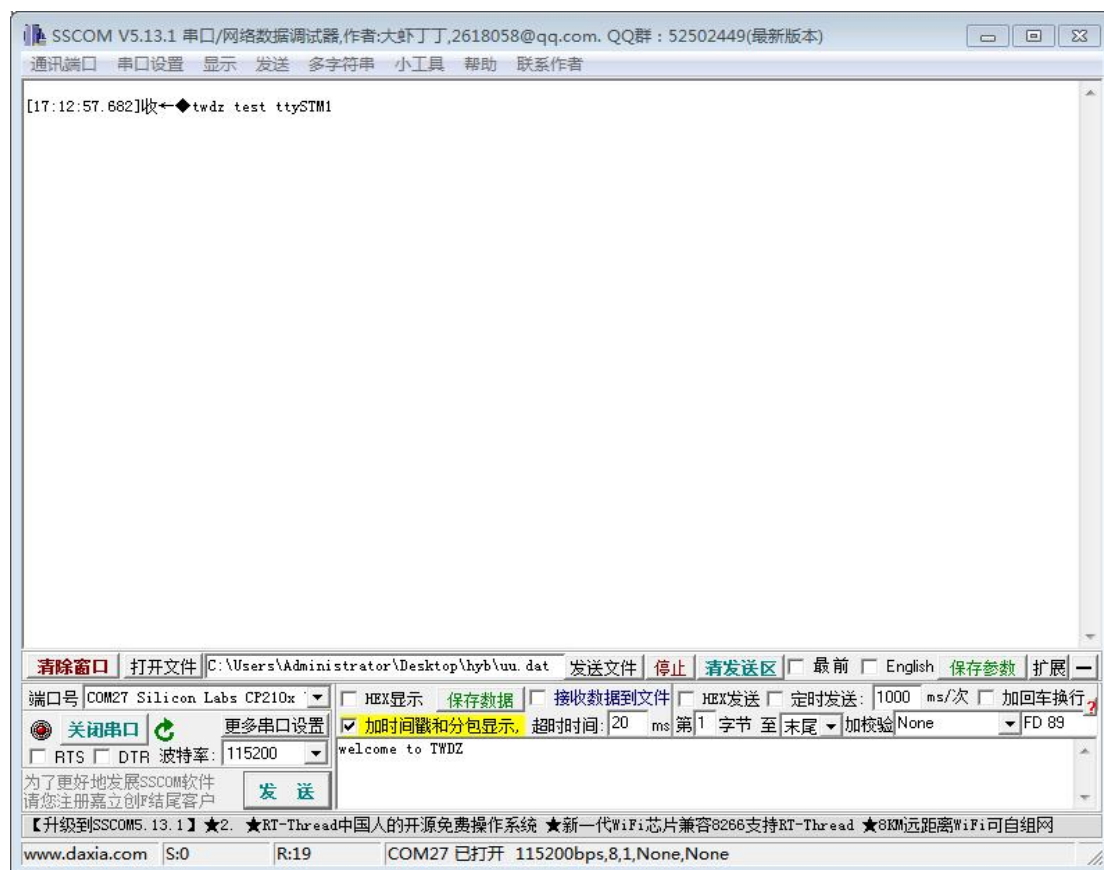
在该对话框中设置主机侧的串口号,并设置串口通讯参数为:波特率 115200,8 位数据位,1 位停止位,无数据校验位。勾选“HEX 显示”选项。设置完成后,点击“打开串口”按钮。

测试开发板发送数据是否正常。方法如下:

1. 在开发板执行如下命令:

```
[root@TWDZ-stm32mp157]:~# echo "twdz test ttySTM1" > /dev/ttySTM1
```

2. 此时 SSCom.exe 会收到/dev/ttySTM1 发过来的信息,如下图所示



接下来测试开发板串口接收数据是否正常。方法如下：

1. 执行以下指令

```
[root@TWDZ-stm32mp157]:~# microcom -t 10000 -s 115200 /dev/ttySTM1
```

####microcom 读数据，10 秒无数据，退出

2. 在 SSCOM.exe 上，输入需要发送的数据，如果数据是二进制字节数据（即非 ASCII 字符）需勾选“HEX 发送”选项。设置完成后，点击“发送”按钮。如下图所示：



3. 查看串口接收信息，会出现

```
[root@TWDZ-stm32mp157]:~# microcom -t 10000 -s 115200 /dev/ttySTM1  
welcome to TWDZ
```

5.4 WIFI 测试

查看是否加在到 wifi 驱动

```
[root@TWDZ-stm32mp157]:~# 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:58  
  
eth0      Link encap:Ethernet Hwaddr ae:18:4d:a0:6b:0d  
          inet addr:192.168.1.10 Bcast:192.168.1.255 Mask:255.255.255.0  
          UP BROADCAST MULTICAST DYNAMIC 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:55 Base address:0xa000  
  
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:2 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:140 (140.0 B) TX bytes:140 (140.0 B)  
  
vlan0     Link encap:Ethernet Hwaddr 68:b9:d3:cf:ca:b2  
          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)
```

1、使能 wifi

```
[root@TWDZ-stm32mp157]:~# rfkill unblock all
```

2、打开 wlan0

```
[root@TWDZ-stm32mp157]:~# ifconfig wlan0 up
```

3、配置 wifi 连接文件

根文件系统的/etc 目录下存在“wpa_supplicant.conf”的配置文件，此文件用于配置要连接的 WIFI 热点以及 WIFI 密码，比如我要连接到“TWDZ”这个热点上。

因此 wpa_supplicant.conf 修改部分文件内容如下所示：

```
[root@TWDZ-stm32mp157]:~#vi /etc/wpa_supplicant.conf
network={
    ssid="TWDZ"
    psk="twdz123456"
}
```

4、连接 wifi

准备好以后就可以使用 wpa_supplicant 工具让 RTL8723DS USB WIFI 连接到热点上，输入如下命令：

```
[root@TWDZ-stm32mp157]:~#wpa_supplicant -D wext -B -i wlan0 -c /etc/wpa_supplicant.conf
```

使用 udhcpc 命令从路由器申请 IP 地址，输入如下命令：udhcpc -i wlan0

```
[root@TWDZ-stm32mp157]:~/wifi# udhcpc -i wlan0
udhcp client (v0.9.8) started
Sending discover...
Sending select for 192.168.43.63...
Lease of 192.168.43.63 obtained, lease time 3599
deleting routers
SIOCDELRT: No such process
adding dns 192.168.43.1
[root@TWDZ-stm32mp157]:~/wifi#
```

可知分配到地址为 192.168.43.63

5、测试 wifi 能否连接外网

```
[root@TWDZ-stm32mp157]:~#ping baidu.com
```

```
[root@TWDZ-stm32mp157]:~/wifi# ping baidu.com
PING baidu.com (39.156.69.79) 56(84) bytes of data.
64 bytes from 39.156.69.79: icmp_seq=1 ttl=48 time=98.8 ms
64 bytes from 39.156.69.79: icmp_seq=2 ttl=48 time=134 ms
^C64 bytes from 39.156.69.79: icmp_seq=3 ttl=48 time=424 ms

--- baidu.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 10354ms
rtt min/avg/max/mdev = 98.850/219.320/424.342/145.712 ms
```

5.5 4G 测试

1. 首先进入\home 下的 4g 目录，直接执行./4g.sh

```
[root@TWDZ-stm32mp157]:/# cd home/4g/
```

```
[root@TWDZ-stm32mp157]:/home/4g# ./4g.sh
```

会出现以下打印信息。

```

abort on (\nBUSY\r)
abort on (\nNO ANSWER\r)
abort on (\nRINGING\r\n\r\nRINGING\r)
timeout set to 40 seconds
send (AT^M)
expect (OK)
AT^M^M
OK
-- got it

send (ATS0=0^M)
expect (OK)
^M
ATS0=0^M^M
OK
-- got it

send (ATE0V1^M)
expect (OK)
^M
ATE0V1^M^M
OK
-- got it

send (AT+CSQ^M)
expect (OK)
^M
^M
+CSQ: 26,99^M
^M
OK
-- got it

send (AT+CGDCONT=1,"IP","CMNET"^M)
expect (OK)
^M
^M
OK
-- got it

send (ATDT*99***1#^M)

```

```

send (ATDT*99***1#^M)
expect (CONNECT)
^M
^M
CONNECT
-- got it

Script chat -s -v -f /etc/ppp/gprs-connect-chat finished (pid 1076), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0xc25a1a30> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth chap MD5> <magic 0x7c10622> <pcomp> <accomp>]
No auth is possible
sent [LCP ConfReq id=0x0 <auth chap MD5>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0xc25a1a30> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x7c10622> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x7c10622> <pcomp> <accomp>]
sent [CCP ConfReq id=0x1 <deflate 15> <deflate(old#) 15> <bsd v1 15>]
sent [IPCP ConfReq id=0x1 <compress VJ 0f 01> <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [LCP DiscReq id=0x2 magic=0x7c10622]
rcvd [LCP ProtRej id=0x3 80 fd 01 01 00 0f 1a 04 78 00 18 04 78 00 15 03 2f]
Protocol-Reject for 'Compression Control Protocol' (0x80fd) received
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1 <compress VJ 0f 01>]
sent [IPCP ConfReq id=0x2 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfNak id=0x2 <addr 10.38.26.11> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
sent [IPCP ConfReq id=0x3 <addr 10.38.26.11> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
rcvd [IPCP ConfAck id=0x3 <addr 10.38.26.11> <ms-dns1 221.179.38.7> <ms-dns2 120.196.165.7>]
Could not determine remote IP address: defaulting to 10.64.64.64
local IP address 10.38.26.11
remote IP address 10.64.64.64
primary DNS address 221.179.38.7
secondary DNS address 120.196.165.7

```


2. 使用指令 ifconfig 指令查看是否生成 ppp0 节点

```
eth0      Link encap:Ethernet  Hwaddr 9E:11:2F:FF:9D:01
          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:55

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:82 errors:0 dropped:0 overruns:0 frame:0
          TX packets:82 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6220 (6.0 KiB)  TX bytes:6220 (6.0 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.38.26.11  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:192 (192.0 B)  TX bytes:545 (545.0 B)

usb0      Link encap:Ethernet  Hwaddr 7A:0A:9C:41:4D:2D
          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)
```

3. 通过指令尝试连接外网 Ping baidu.com

```
PING baidu.com (39.156.69.79) 56(84) bytes of data:
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=5 ttl=50 time=76.2 ms
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=6 ttl=50 time=80.4 ms
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=7 ttl=50 time=78.7 ms
64 bytes from 39.156.69.79 (39.156.69.79): icmp_seq=8 ttl=50 time=76.7 ms
^C
--- baidu.com ping statistics ---
8 packets transmitted, 4 received, 50% packet loss, time 11283ms
rtt min/avg/max/mdev = 76.177/77.981/80.383/1.667 ms
```

5.6 时钟设置

Linux 将时钟分为系统时钟(System Clock)和硬件时钟(Real Time Clock, 简称 RTC)两种。系统时钟是由 Linux 内核所维护的时钟, 用户一般使用和看到的都是系统时钟。而硬件时钟则是由主板上的电池供电的主板硬件时钟。系统时钟在系统断电后即会消失, 但 RTC 时钟在主板电池有电的情况下会长期运行。因此每次上电时, Linux 内核都会读取主板上的 RTC 时钟, 并将它同步到系统时钟。下面列出一些与时钟相关的命令:

5.6.1 查看系统时钟:

使用 date 命令可以查看系统时钟:

```
[root@TWDZ-stm32mp157]:~# date
Sat Jan  4 21:09:19 CST 2020
```

5.6.2 查看 RTC 时钟:

使用 hwclock 命令可以查看 RTC 时钟:

```
[root@TWDZ-stm32mp157]:~# hwclock
Mon Dec  7 10:01:30 2020  0.000000 seconds
```

5.6.3 设置 RTC 时钟:

使用 hwclock -w, 可以将系统时钟写入 RTC 时钟:

```
[root@TWDZ-stm32mp157]:~# hwclock -w
```

5.6.4 同步系统时钟：

使用 `hwclock -s`，可以将 RTC 时钟写入系统时钟：

```
[root@TWDZ-stm32mp157]:~# hwclock -s
```

通过上面的叙述可以看出，如果想要改变当前的系统时间，且希望系统重启后改变依然生效，需要执行如下两步操作：

- 使用 `date -s` 命令修改当前的系统时钟；
- 使用 `hwclock -w` 命令将修改后的系统时钟写入 RTC 时钟。

例如需要将当前时钟设置为 2020-12-07 10: 03: 10，并希望该改变在系统重启后依然有效，应执行如下命令：

```
[root@TWDZ-stm32mp157]:~# date -s "2020-12-07 10:03:10"
```

```
Mon Dec 7 10:03:10 UTC 2020
```

```
[root@TWDZ-stm32mp157]:~# hwclock -w
```

重启计算机后，如果 RTC 正常的话，使用以下命令可以查看到刚刚设定的时间值：

```
[root@TWDZ-stm32mp157]:~# hwclock
```

5.7 CAN 测试

TW-STM32MP157-EVM 底板上只有一路 CAN 接口，如果想要测试 CAN 接口，用户需要找另外的 CAN 口进行对接测试。

1. 设置 CAN 的速率

```
[root@TWDZ-stm32mp157]:~# ip link set can0 up type can bitrate 50000
```

2. 发送 CAN 数据

```
[root@TWDZ-stm32mp157]:~# cansend can0 123#01.02.03.04.05.06
```

3. 接收 CAN 数据

```
[root@TWDZ-stm32mp157]:~# candump -ta can0
```

5.8 摄像头测试

注：在测试之前请接好相应的摄像头，注意引脚顺序，执行命令后就可以在液晶上看到相应的摄像头画面，TW-STM32MP157-EVM 开发板提供 CSI 摄像头接口：

在控制端输入 `v4l2-ctl --device=/dev/video0 --list-formats-ext` 查看摄像头支持格式、分辨率及帧率。

```
[root@TWDZ-stm32mp157]:~# v4l2-ctl --device=/dev/video0 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
Type: Video Capture

[0]: 'JPEG' (JFIF JPEG, compressed)
    Size: Discrete 176x144
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 320x240
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 640x480
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
        Interval: Discrete 0.017s (60.000 fps)
    Size: Discrete 720x480
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 720x576
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 1024x768
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 1280x720
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 1920x1080
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 2592x1944
        Interval: Discrete 0.067s (15.000 fps)
[1]: 'UYVY' (UYVY 4:2:2)
    Size: Discrete 176x144
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 320x240
        Interval: Discrete 0.067s (15.000 fps)
        Interval: Discrete 0.033s (30.000 fps)
    Size: Discrete 640x480
        Interval: Discrete 0.067s (15.000 fps)
```

为了测试 CSI 接口, 需要打开 weston 服务, 在命令行下执行如下命令, 可以将采集图像显示于 LCD 屏:

```
[root@TWDZ-stm32mp157]:/#systemctl stop qtdesktop
[root@TWDZ-stm32mp157]:/#systemctl start weston@root.service
[root@TWDZ-stm32mp157]:/# gst-launch-1.0 v4l2src device=/dev/video0 ! "video/x-raw,
format=YUY2, width=640, height=480, framerate=(fraction)60/1" ! waylandsink
```

5.9 网络测试

TW-STM32MP157-EVM 评估板有一路百兆以太网接口, 使用标准的 RJ45 网口插座, 插座内带状态指示灯。

可使用 ifconfig 指令来显示或者配置网络, 查看网络信息。

```
[root@TWDZ-stm32mp157]:/# ifconfig
```



```
[root@TWDZ-stm32mp157]:/# ifconfig
eth0      Link encap:Ethernet  Hwaddr 66:a9:22:26:bb:3b
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST DYNAMIC MTU:1500 Metric:1
          RX packets:640 errors:0 dropped:49 overruns:0 frame:0
          TX packets:35 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:57494 (56.1 KiB)  TX bytes:3300 (3.2 KiB)
          Interrupt:55 Base address:0xa000

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:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:732 (732.0 B)  TX bytes:732 (732.0 B)
```

插上网线到以太网接口处可以看到如下信息，系统自动获取了 ip 同理。

```
[root@TWDZ-stm32mp157]:~# [13952.809184] stm32-dwmac 5800a000.ethernet eth0: Link is
Up - 1Gbps/Full - flow control rx/tx
```

```
[root@TWDZ-stm32mp157]:/# udhcpc -i eth0
```

```
[root@TWDZ-stm32mp157]:/# ifconfig
eth0      Link encap:Ethernet  Hwaddr 66:a9:22:26:bb:3b
          inet addr:192.168.0.123  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST DYNAMIC MTU:1500 Metric:1
          RX packets:4256 errors:0 dropped:181 overruns:0 frame:0
          TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:353277 (344.9 KiB)  TX bytes:9767 (9.5 KiB)
          Interrupt:55 Base address:0xa000

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:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:732 (732.0 B)  TX bytes:732 (732.0 B)
```

如果对应网卡没有自动获取到 IP，请使用下面的指令获取。“-i”是指定网卡名称，如不指定，会使用默认会使用 eth0。

```
[root@TWDZ-stm32mp157]:/# ifconfig eth0 down // 关闭网口,网卡名字请根据实际情况
修改, down 表示关闭网口
```

```
[root@TWDZ-stm32mp157]:/# ifconfig eth0 up // 打开网口,网卡名字请根据实际情况
修改, up 表示打开网口
```

测试网口是否能上网，以访问 www.baidu.com 为例，执行如下命令，“-I”代表指定网口，不加“-I”则使用默认网卡（默认网卡指的是有网络接入的一端，如果两个网口都有网络接入，则使用 eth0 作为默认网卡）。按“Ctrl+c”终止 ping 指令。百度的实际地址根据网络运营商不同，访问的地址会不同。

```
[root@TWDZ-stm32mp157]:/# ping baidu.com
```

```
[root@TWDZ-stm32mp157]:/# ping baidu.com
PING baidu.com (220.181.38.148) 56(84) bytes of data.
64 bytes from 220.181.38.148: icmp_seq=1 ttl=53 time=42.3 ms
64 bytes from 220.181.38.148: icmp_seq=2 ttl=53 time=42.4 ms
64 bytes from 220.181.38.148: icmp_seq=3 ttl=53 time=42.4 ms
64 bytes from 220.181.38.148: icmp_seq=4 ttl=53 time=42.2 ms
--- baidu.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 15154ms
rtt min/avg/max/mdev = 42.266/42.360/42.452/0.164 ms
[root@TWDZ-stm32mp157]:/#
```

使用 route 命令查看网关后，并 ping 网关。

```
[root@TWDZ-stm32mp157]:/# route
```

```
[root@TWDZ-stm32mp157]:/# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 192.168.0.1 0.0.0.0 UG 0 0 0 eth0
192.168.0.0 * 255.255.255.0 U 0 0 0 eth0
```

由上可知网关为 gao.ke，根据路由器不同，网关可能不同。ping 网关可测试内网与开发板连接是否正常。下面指令不加“-I”参数，使用默认网卡。

```
[root@TWDZ-stm32mp157]:/# ping gao.ke
```

```
[root@TWDZ-stm32mp157]:/# ping gao.ke
PING g.17986.net (104.160.174.162) 56(84) bytes of data.
64 bytes from MAYALINZTIME.NET (104.160.174.162): icmp_seq=2 ttl=54 time=159 ms
64 bytes from 162.174.160.104.in-addr.arpa (104.160.174.162): icmp_seq=3 ttl=54
time=160 ms
64 bytes from 162.174.160.104.in-addr.arpa (104.160.174.162): icmp_seq=4 ttl=54
time=159 ms
AC
--- g.17986.net ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3019ms
rtt min/avg/max/mdev = 159.871/159.936/160.019/0.465 ms
```

5.10 音频测试

开始录音 5 秒钟，执行下面的指令：

```
[root@TWDZ-stm32mp157]:/# arecord -r 44100 -f S16_LE -d 5 record.wav
```

指令解释：

- (1) -r 44100: 采样率 44.1K
- (2) -f S16_LE: 以 S16_LE 格式采样
- (3) -d 10: 录音长度 10s
- (4) record.wav: 录音存生成的音频文件

播放录制的音频文件，执行下面的指令：

```
[root@TWDZ-stm32mp157]:/# aplay record.wav
```

5.11 TF 卡测试

将 TF 卡插入到开发板 TF 卡插槽中，Linux 操作系统会检测到 TF 卡，并在控制台终端上打印 TF 卡的相关信息，例如：

```
[root@TWDZ-stm32mp157]:/# [15376.081750] mmc1: host does not support reading read-only
```

```
switch, assuming write-enable
```

```
[15376.093354] mmc1: new high speed SDHC card at address b368
```

```
[15376.101728] mmcblk1: mmc1:b368 SD08G 7.42 GiB
```

```
[15376.108947] mmcblk1: p1 p2
```

```
[15376.767721] EXT4-fs (mmcblk1p2): recovery complete
```

```
[15376.771141] EXT4-fs (mmcblk1p2): mounted filesystem with ordered data mode. Opts: (null)
```

```
[15376.799782] FAT-fs (mmcblk1p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

TF 卡被挂载到/run/media/mmcblk1p1 目录下,为了将 TF 卡根目录上的 test 文件拷贝到/home 目录下,可执行如下命令:

```
[root@TWDZ-stm32mp157]:/#cp /run/media/mmcblk1p1/test /home
```

为了将/home 目录下的 test1 文件拷贝到 TF 卡根目录上,可执行如下命令:

```
[root@TWDZ-stm32mp157]:/#cp /home/test1 /run/media/mmcblk1p1
```

此处假定 TF 卡被挂载在/run/media/mmcblk2p1 目录下。umount 会确保所有缓存的数据都被正确的写入 TF 卡。在 umount 成功后,即可拔出 TF 卡。在调用 umount 前,必须确保 TF 卡上的文件没有被其他程序所占用且用户当前的工作目录不在 TF 卡的挂载目录上,否则调用 umount 会提示失败,如下所示:

```
umount: /run/media/mmcblk1p1/: target is busy (In some cases useful info about processes that use the device is found by lsof(8) or fuser(1)).
```

5.12 U 盘使用

将格式为 FAT32 的 U 盘插入到开发板 USB HOST 接口上,Linux 操作系统会检测到 U 盘,并在控制台终端上打印 U 盘的相关信息,例如:

```
[root@TWDZ-stm32mp157]:~# usb 1-1.2: new high-speed USB device number 4 using ci_hdrc
```

```
usb-storage 1-1.2:1.0: USB Mass Storage device detected
```

```
scsi host0: usb-storage 1-1.2:1.0
```

```
scsi 0:0:0:0: Direct-Access      SMI          USB DISK          1100 PQ: 0 ANSI: 4
```

```
sd 0:0:0:0: [sda] 15769600 512-byte logical blocks: (8.07 GB/7.51 GiB)
```

```
sd 0:0:0:0: [sda] Write Protect is off
```

```
sd 0:0:0:0: [sda] No Caching mode page found
```

```
sd 0:0:0:0: [sda] Assuming drive cache: write through
```

```
sda: sda1
```

```
sd 0:0:0:0: [sda] Attached SCSI removable disk
```

```
FAT-fs (sda1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

在此例中,从打印信息可以看出 Linux 操作系统检测到一个容量为 8GB 的 U 盘,其对应的设备名为 sda。

系统会在/etc 目录下为 U 盘的每个分区都生成一个目录,目录的名字为 sdxn (x 用于区分不同的 U 盘,n 用于区分不同的分区,x=a、b、c…… n=1、2、3……)。U 盘的每个分区就挂载在这些目录下。U 盘挂载成功后,即可对 U 盘进行文件查看、文件拷贝等操作。以下以文件拷贝操作为例,介绍 U 盘的使用。

假定 U 盘被挂载到/run/media/sda1 目录下,为了将 U 盘根目录上的 test 文件拷贝到/home 目录下,可执行如下命令:

```
[root@TWDZ-stm32mp157]:/# cp /run/media/sda1/test /home
```

为了将/home 目录下的 test1 文件拷贝到 U 盘根目录上,可执行如下命令:

```
[root@TWDZ-stm32mp157]:/# cp /home/test1 /run/media/sda1
```

此处假定 U 盘被挂载在/run/media/sda1 目录下。umount 会确保所有缓存的数据都被正确的写入 U 盘。在 umount 成功后,即可拔出 U 盘。在调用 umount 前,必须确保 U 盘上的文件没有被其他程序所占用且用户当前的工作目录不在 U 盘的挂载目录上,否则调用 umount 会提示失败,如下所示:

```
umount: /run/media/sda1/: target is busy(In some cases useful info about processes that use the device is found by lsof(8) or fuser(1).)
```

5.13 USB 鼠标与 USB 键盘使用

将 USB 鼠标插入到开发板 USB HOST 接口上, Linux 操作系统会检测到 USB 鼠标,并在控制台终端上打印 USB 鼠标的相关信息,例如:

```
usb 1-1.2: new low-speed USB device number 4 using ci_hdrc
input: USB Optical Mouse as
/devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1.2/1-1.2: 1.0/0003:
1BCF: 0007.0001/input/ input2
hid-generic 0003: 1BCF: 0007.0001: input: USB HID v1.10 Mouse [USB Optical Mouse] on
usb-ci_hdrc.1-1.2/input0
```

将 USB 键盘插入到开发板 USB HOST 接口上, Linux 操作系统会检测到 USB 键盘,并在控制台终端上打印 USB 键盘的相关信息,例如:

```
usb 1-1.2: new low-speed USB device number 6 using ci_hdrc
input: USB USBKeyboard as
/devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1.2/1-1.2 : 1.0/0003 :
1A2C: 0002.0004/input/ input5
hid-generic 0003: 1A2C: 0002.0004: input: USB HID v1.10 Keyboard [USB USBKeyboard]
on usb-ci_hdrc.1-1.2/input0
input: USB USBKeyboard as
/devices/soc0/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1.2/1-1.2 : 1.1/0003 :
1A2C: 0002.0005/input/ input6
hid-generic 0003: 1A2C: 0002.0005: input: USB HID v1.10 Device [USB USBKeyboard]
on usb-ci_hdrc.1-1.2/input1
```


5.14 LCD 背景亮度调节

LCD 屏幕的背光支持 8 级变化，亮度级数为 0~7。可以通过以下命令查看等级：

```
[root@TWDZ-stm32mp157]:~#cat /sys/class/backlight/panel-backlight/max_brightness // 查看
lcd 最大亮度等级
```

修改当前屏幕背光亮度等级，执行命令如下：

```
[root@TWDZ-stm32mp157]:~# echo 5 > /sys/class/backlight/panel-backlight/brightness
```

5.15 USB 接口测试

TW-AC6G-EVM 评估板有 2 路 USB2.0 HOST 接口 CN20，使用标准的双层 USB-A 插座。1 路 USB2.0 HOST 接口 CZ1 和 CN21 二选一使用，CZ1 使用标准的 PH2.0-4A 插座，CN21 使用侧插 USB-AF 插座，方便用户扩展接口。

TW-AC6G-EVM 评估板有 1 路 Micro_USB 接口 CN38，使用标准的 Micro_USB 插座，该接口默认为 Device 接口。

TW-AC6G-EVM 评估板 USB/OTG 接口说明：

- ◆ USB_HOST1~USB_HOST3 为 HOST 模式，默认为 HOST 模式；
- ◆ USB1_HOST 支持 HOST 模式；USB_OTG1 支持切换为 HOST/DEVICE

以下测试根据使用的 U 盘的不同和实验环境不同，测试结果会有所差异。将 FAT32 格式 U 盘插到开发板 USB_HOST1~USB_HOST3/USB1_HOST 其中一个接口。

插入后会打印如下信息，可以从中看到 U 盘大小和挂载名，如下图所示：

```
[root@TWDZ-stm32mp157]:~# [16133.817192] usb 2-1.5: new high-speed USB device number 5 using ehci-platform
[16133.888659] usb-storage 2-1.5:1.0: USB Mass Storage device detected
[16133.905029] scsi host0: usb-storage 2-1.5:1.0
[16134.968808] scsi 0:0:0:0: Direct-Access Kingston DataTraveler 3.0 PQ: 0 ANSI: 6
[16134.978595] sd 0:0:0:0: Attached scsi generic sg0 type 0
[16134.985979] sd 0:0:0:0: [sda] 60437492 512-byte logical blocks: (30.9 GB/28.8 GiB)
[16134.998463] sd 0:0:0:0: [sda] write Protect is off
[16135.002664] sd 0:0:0:0: [sda] write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[16135.055013] sda: sda1
[16135.069957] sd 0:0:0:0: [sda] Attached SCSI removable disk
[16135.359988] udevd[3499]: failed to execute '/etc/mount-usb.sh' '/etc/mount-usb.sh sda1': No such file or directory
[16135.611909] FAT-fs (sda1): volume was not properly unmounted. Some data may be corrupt. Please run fsck.
```

输入 df -h 查看 U 盘的挂载路径，可以看到下图 U 盘已经挂载在 /run/media/ 下，挂载名为 sda1。

```
[root@TWDZ-stm32mp157]:~# df
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/root        1277484 1000592    193948   84% /
devtmpfs         379092         4    379088    1% /dev
tmpfs            445140        188    444952    1% /run
tmpfs            445140        420    444720    1% /var/volatile
/dev/mmcblk2p2    8887         8282         0 100% /run/media/mmcblk2p2
/dev/mmcblk1p2   1426656 1410272         0 100% /run/media/mmcblk1p2
/dev/mmcblk1p1   511720  406296    105424   80% /run/media/mmcblk1p1
/dev/sda1       30203008 5567568  24635440   19% /run/media/sda1
```

写速度测试：

输入 `time dd if=/dev/zero of=/run/media/sda1/test bs=1024k count=100 conv=fdatasync`

```
[root@TWDZ-stm32mp157]:~# time dd if=/dev/zero of=/run/media/sda1/test bs=1024k count=100 conv=fdatasy
nc
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 11.3435 s, 9.2 MB/s
real    0m 11.39s
user    0m 0.00s
sys     0m 2.30s
[root@TWDZ-stm32mp157]:~#
```

本次写 100MiB，速度为 8.8MB/s。

5.16 触摸屏测试

测试触摸是否有反应

```
[root@TWDZ-stm32mp157]:~# hexdump /dev/input/event0
```

```
[root@TWDZ-stm32mp157]:~# hexdump /dev/input/event0
00000000 9966 5e10 d1f3 000d 0003 0035 018f 0000
00000010 9966 5e10 d1f3 000d 0003 0036 0100 0000
00000020 9966 5e10 d1f3 000d 0003 0000 018f 0000
00000030 9966 5e10 d1f3 000d 0003 0001 0100 0000
00000040 9966 5e10 d1f3 000d 0000 0000 0000 0000
00000050 9966 5e10 ca36 000e 0003 0035 018b 0000
00000060 9966 5e10 ca36 000e 0003 0036 00ff 0000
00000070 9966 5e10 ca36 000e 0003 0000 018b 0000
00000080 9966 5e10 ca36 000e 0003 0001 00ff 0000
00000090 9966 5e10 ca36 000e 0000 0000 0000 0000
000000a0 9966 5e10 f331 000e 0003 0035 0186 0000
000000b0 9966 5e10 f331 000e 0003 0000 0186 0000
000000c0 9966 5e10 f331 000e 0000 0000 0000 0000
000000d0 9966 5e10 1c39 000f 0003 0035 017f 0000
000000e0 9966 5e10 1c39 000f 0003 0036 00fe 0000
000000f0 9966 5e10 1c39 000f 0003 0000 017f 0000
0000100 9966 5e10 1c39 000f 0003 0001 00fe 0000
0000110 9966 5e10 1c39 000f 0000 0000 0000 0000
0000120 9967 5e10 0642 0000 0003 0035 0178 0000
```

5.17 CPU 温度

CPU 温度的高低与环境温度及 CPU 运行状况有关，例如跑一些大的应用，跑视频音频等，温度会高一些，建议不要用手去触摸 CPU，以免造成损坏。使用以下命令可以读出 CPU 温度。

```
cat /sys/class/hwmon/hwmon0/temp1_input
```

```
[root@TWDZ-stm32mp157]:~# cat /sys/class/hwmon/hwmon0/temp1_input
53127
```

5.18 CPU 主频

使用以下命令 `cpufreq-info` 测试主频。


```
[root@TWDZ-stm32mp157]:~# cpufreq-info
cpufrequtils 008: cpufreq-info (C) Dominik Brodowski 2004-2009
Report errors and bugs to cpufreq@vger.kernel.org, please.
analyzing CPU 0:
  driver: cpufreq-dt
  CPUs which run at the same hardware frequency: 0 1
  CPUs which need to have their frequency coordinated by software: 0 1
  maximum transition latency: 0.00 ms.
  hardware limits: 650 MHz - 650 MHz
  available frequency steps: 650 MHz
  available cpufreq governors: ondemand, performance, schedutil
  current policy: frequency should be within 650 MHz and 650 MHz.
    The governor "ondemand" may decide which speed to use
    within this range.
  current CPU frequency is 650 MHz (asserted by call to hardware).
  cpufreq stats: 650 MHz:100.00%
analyzing CPU 1:
  driver: cpufreq-dt
  CPUs which run at the same hardware frequency: 0 1
  CPUs which need to have their frequency coordinated by software: 0 1
  maximum transition latency: 0.00 ms.
  hardware limits: 650 MHz - 650 MHz
  available frequency steps: 650 MHz
  available cpufreq governors: ondemand, performance, schedutil
  current policy: frequency should be within 650 MHz and 650 MHz.
    The governor "ondemand" may decide which speed to use
    within this range.
  current CPU frequency is 650 MHz (asserted by call to hardware).
  cpufreq stats: 650 MHz:100.00%
```

由上图可得知,CPU 主频工作在 650MHZ, 调频模式为 “ondemand”, 也可以通过以下指令查看当前模式和当前的频率。

```
[root@TWDZ-stm32mp157]:~# cd /sys/devices/system/cpu/cpufreq/policy0/
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_cur_freq
650000
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_max_freq
650000
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_min_freq
650000
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_governor
ondemand
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_cur_freq
400000
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_max_freq
800000
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_min_freq
400000
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0# cat scaling_governor
ondemand
[root@TWDZ-stm32mp157]:/sys/devices/system/cpu/cpufreq/policy0#
```

5.19 查看 CPU 信息

```
[root@TWDZ-stm32mp157]:~# cat /proc/cpuinfo
```

```
[root@TWDZ-stm32mp157]:~# cat /proc/cpuinfo
processor       : 0
model name     : ARMv7 Processor rev 5 (v7l)
BogoMIPS      : 24.00
Features      : half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstr
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xc07
CPU revision   : 5

processor       : 1
model name     : ARMv7 Processor rev 5 (v7l)
BogoMIPS      : 24.00
Features      : half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstr
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part       : 0xc07
CPU revision   : 5

Hardware       : STM32 (Device Tree Support)
Revision      : 0000
Serial        : 001f00263030510332313933
[root@TWDZ-stm32mp157]:~#
```

5.20 查看内存信息

```
[root@TWDZ-stm32mp157]:~# cat /proc/meminfo
```

```
[root@TWDZ-stm32mp157]:~# cat /proc/meminfo
MemTotal:      953304 kB
MemFree:       747736 kB
MemAvailable:  850444 kB
Buffers:       14304 kB
Cached:        98708 kB
SwapCached:    0 kB
Active:        67064 kB
Inactive:      83512 kB
Active(anon):  38252 kB
Inactive(anon): 8536 kB
Active(file):  28812 kB
Inactive(file): 74976 kB
Unevictable:   0 kB
Mlocked:       0 kB
HighTotal:     262140 kB
HighFree:      119300 kB
LowTotal:      691164 kB
LowFree:       628436 kB
SwapTotal:     0 kB
SwapFree:      0 kB
Dirty:         48 kB
Writeback:     0 kB
AnonPages:     37560 kB
Mapped:        44240 kB
Shmem:         9228 kB
KReclaimable:  17548 kB
Slab:          42304 kB
SReclaimable:  17548 kB
SUnreclaim:    24756 kB
KernelStack:   1000 kB
PageTables:    1264 kB
NFS_Unstable:  0 kB
Bounce:        0 kB
WritebackTmp:  0 kB
CommitLimit:   476652 kB
Committed_AS:  378448 kB
VmallocTotal:  245760 kB
VmallocUsed:    2604 kB
VmallocChunk:   0 kB
PerCpu:        400 kB
CmaTotal:      131072 kB
CmaFree:       115352 kB
[root@TWDZ-stm32mp157]:~#
```

6. 环境搭建

6.1 安装虚拟机软件 VMware

安装 Ubuntu 的前提是什么？需要一个虚拟机 VMware 去启动它，虚拟机顾名思义就是虚拟出来的一个机器，然后在这个机器上安装任何你想要的系统，相当于在克隆出一个你的电脑，这样的话，主机上运行 Window 系统，当我们需要用到 Ubuntu 的话，打开安装有 Ubuntu 系统的虚拟机就可以了。

Vmware Workstation 软件可以在 Wmware 官网下载，下载地址：<https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>。下载最新版的 Vmware Workstation Pro 15。

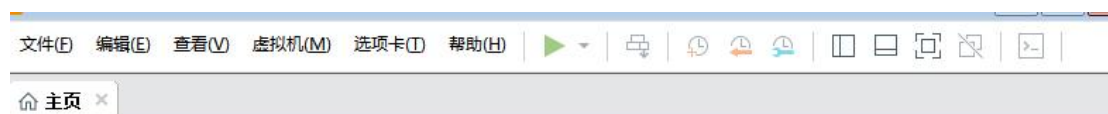
VMware 虚拟机的版本高低可能会导致 Ubuntu 无法正常开启。需要去修改光盘资料所提供 Ubuntu 的 Ubuntu 64 位.vmx 文件。



通过修改.vmx 文件里面的相应参数，如下图所示。

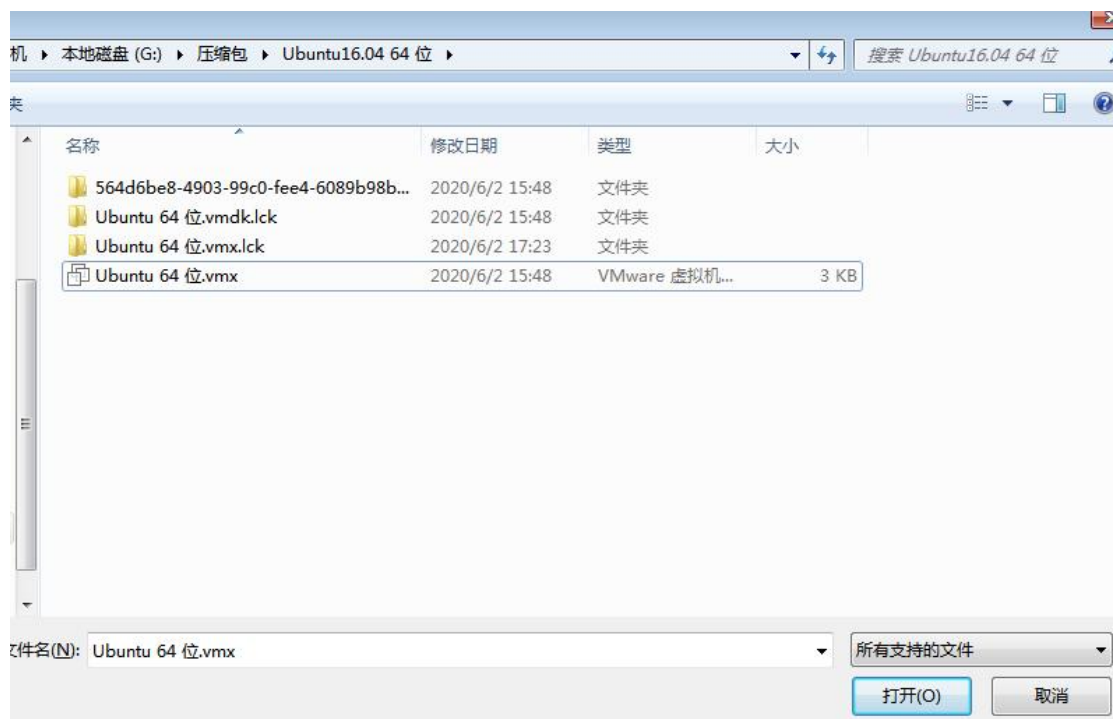
```
encoding = "GBK"
config.version = "8"
virtualHW.version = "16"
mks.enable3d = "TRUE"
pciBridge0.present = "TRUE"
pciBridge4.present = "TRUE"
pciBridge4.virtualDev = "pcieRootPort"
pciBridge4.functions = "8"
pciBridge5.present = "TRUE"
pciBridge5.virtualDev = "pcieRootPort"
pciBridge5.functions = "8"
pciBridge6.present = "TRUE"
pciBridge6.virtualDev = "pcieRootPort"
pciBridge6.functions = "8"
pciBridge7.present = "TRUE"
pciBridge7.virtualDev = "pcieRootPort"
pciBridge7.functions = "8"
vmci0.present = "TRUE"
hpet0.present = "TRUE"
displayName = "Ubuntu 64"
guestOS = "ubuntu-64"
nvram = "Ubuntu 64.nvram"
virtualHW.productCompatibility = "hosted"
powerType.powerOff = "soft"
```

安装完毕后，打开 VMware Workstation 软件，打开虚拟机，找到虚拟机解压的路径。



WORKSTATION 15.5 PRO™





打开虚拟机，进入系统



6.2 交叉工具链

光盘资料中提供的 Ubuntu 已经把交叉工具链的环境搭建好了，用户可以直接进行使用。

Ubuntu 版本	16.04
交叉工具链版本	arm-none-linux-gnueabi-hf-gcc 9.2.1
交叉工具链的所在路径	/home/twdz/MP1/arm-linux/gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi-hf/bin

TW-STM32MP157 开发板使用 arm-none-linux-gnueabi-hf-gcc 9.2.1 版本交叉编译器，其安装包为 gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi-hf.tar.xz 可从“TW-STM32MP157 光盘资料\2.软件开发参考资料\1.编译工具”中直接获取。

可以简单的测试一下交叉编译器是否可以正常运行。方法如下：

在命令行下执行：

```
$source
/home/twdz/MP1/arm-linux/gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi-hf/bin
$arm-none-linux-gnueabi-hf-gcc -v
```

如果能显示编译器的版本信息，则表明编译器已经可以正常运行了，如下图所示：

```
twdz@ubuntu:~/MP1/kernel$ arm-none-linux-gnueabi-hf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-none-linux-gnueabi-hf-gcc
COLLECT_LTO_WRAPPER=/home/twdz/MP1/arm-linux/gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi-hf/bin/./libexec/gcc/arm-none-linux-gnueabi-hf/9.2.1/lto-wrapper
Target: arm-none-linux-gnueabi-hf
Configured with: /tmp/dgboter/bbs/rhev-vm7--rhe6x86_64/buildbot/rhe6x86_64--arm-none-linux-gnueabi-hf/build/src/gcc/configure --target=arm-none-linux-gnueabi-hf --prefix= --with-sysroot=/arm-none-linux-gnueabi-hf/libc --with-build-sysroot=/tmp/dgboter/bbs/rhev-vm7--rhe6x86_64/buildbot/rhe6x86_64--arm-none-linux-gnueabi-hf/build/build-arm-none-linux-gnueabi-hf/install/arm-none-linux-gnueabi-hf/libc --with-bugurl=https://bugs.linaro.org/ --enable-gnu-indirect-function --enable-shared --disable-libssp --disable-libmudflap --enable-checking=release --enable-languages=c,c++,fortran --with-gmp=/tmp/dgboter/bbs/rhev-vm7--rhe6x86_64/buildbot/rhe6x86_64--arm-none-linux-gnueabi-hf/build/build-arm-none-linux-gnueabi-hf/host-tools --with-mpc=/tmp/dgboter/bbs/rhev-vm7--rhe6x86_64/buildbot/rhe6x86_64--arm-none-linux-gnueabi-hf/build/build-arm-none-linux-gnueabi-hf/host-tools --with-isl=/tmp/dgboter/bbs/rhev-vm7--rhe6x86_64/buildbot/rhe6x86_64--arm-none-linux-gnueabi-hf/build/build-arm-none-linux-gnueabi-hf/host-tools --with-fpu=neon --with-float=hard --with-mode=thumb --with-arch=armv7-a --with-pkgversion='GNU Toolchain for the A-profile Architecture 9.2-2019.12 (arm-9.10)'
Thread model: posix
gcc version 9.2.1 20191025 (GNU Toolchain for the A-profile Architecture 9.2-2019.12 (arm-9.10))
```

6.3 编译 HelloWorld 源程序

在 Linux 宿主机任意目录下创建一个 HelloWorld 文件夹，用于存放 helloworld.c 源文件。此处假定在/home/twdz/demo 下创建 HelloWorld 文件夹。执行的命令如下：

```
$mkdir /home/twdz/demo
$mkdir /home/twdz/demo/HelloWorld
$cd /home/twdz/demo/HelloWorld
```

在该目录下新建一个 helloworld.c 的源文件，其内容如下：

```
#include<stdio.h>
int main()
{
    Printf(“helloworld!\n”);
    Return 0;
```



```
}
```

保存退出后，进入源码所在目录，在命令行下执行如下命令编译程序：

```
arm-none-linux-gnueabi-gcc -o helloworld helloworld.c
```

将交叉编译出来的执行文件 helloworld 下载到开发板上运行。

```
~#chmod a+x helloworld
```

```
~# ./helloworld
```

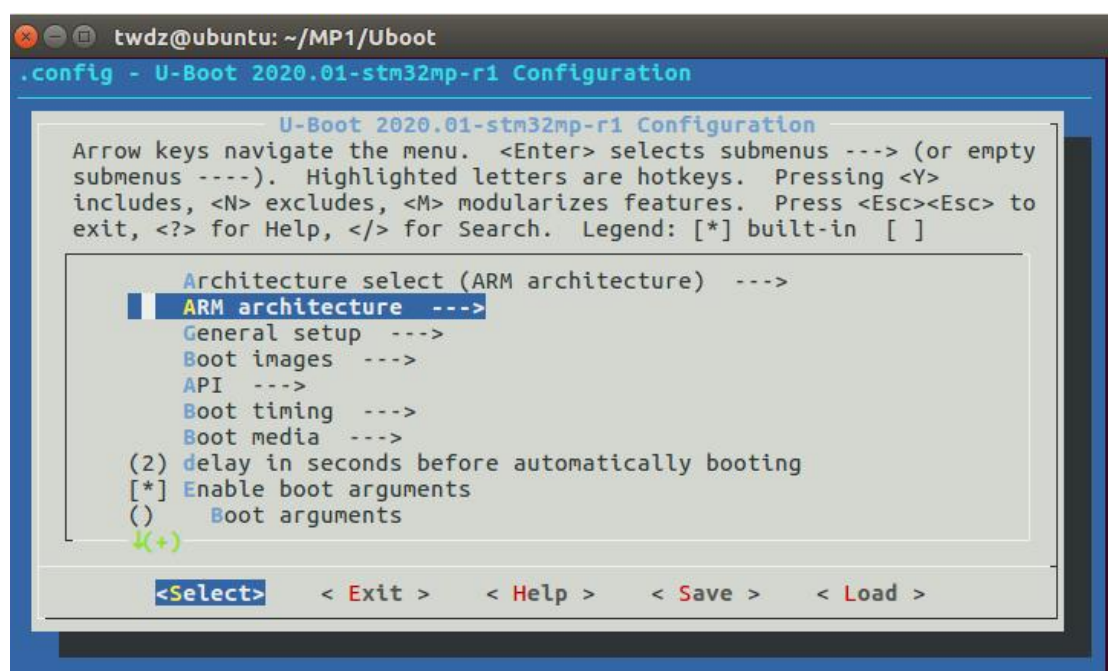
```
helloworld
```

从执行结果可以看出，在终端上已经打印出了“Hello World”字符串。

6.4 编译 Uboot

uboot 的顶层 Makefile 中我们已经定义了 ARCH 和 CROSS_COMPILE 的值，所以后面输入指令的时候可以简化。

通过输入“make menuconfig”来打开图形化配置界面



通过输入“./build_all.sh”执行完后源根目录会出现以下新编译出来的文件。

```
twdz@ubuntu: ~/MP1/Uboot
twdz@ubuntu:~/MP1/Uboot$ ls
api          doc          MAINTAINERS  u-boot.cfg
arch         drivers      Makefile     u-boot.cfg.configs
board        dts          net          u-boot.dtb
build_all.sh env          post         u-boot-dtb.bin
build_config.sh examples     README       u-boot.lds
cmd          fs           scripts      u-boot.map
common       include      System.map   u-boot-nodtb.bin
config.mk    Kbuild       test         u-boot.srec
configs     Kconfig     tools        u-boot.stm32
CONTRIBUTING.md lib          u-boot      u-boot.stm32.log
disk         Licenses    u-boot.bin   u-boot.syn
twdz@ubuntu:~/MP1/Uboot$
```

通过输入 “./build_config.sh” 指令可以恢复默认配置。

6.5 编译内核

内核源码的顶层提供了各种编译执行脚本，用户可以根据不同的需求去执行相应的脚本，得到目标固件。

shell 脚本	说明	固件路径
built-all-tw.sh	编译内核映像、设备树	编译出来的目标文件 源码顶层的 arch/arm/boot/uImage 源码顶层的 arch/arm/boot/dts/stm32mp157a-tw.dtb
built-modules.sh	编译内核模块	需根据在内核配置编译的模块的路径，在源码顶层的 driver/目录下相应的目录，可参照 “错误!未找到引用源。” 这章节
built-menuconfig.sh	使用 menuconfig 配置内核	

6.5.1 内核源码简介

Linux 内核源码很复杂，包含多级目录，形成一个庞大的树状结构，通常称为 Linux 源码目录树。TW-STM32MP157-EVM 开发板使用 Linux5.4.31 内核版本，本节以该版本为例介绍 Linux 源码的目录结构：

目录	说明
arch	包含各体系结构特定的代码，如 arm、x86、ia64、mips 等，在每个体系结构目录下通常都有： —boot 内核需要的特定平台代码 —kernel 体系结构特有的代码 —lib 通用函数在特定体系结构的实现 —math-emu 模拟 FPU 的代码，在 ARM 中，使用 mach-xxx 代替 —mm 特定体系结构的内存管理实现

	—include 特定体系的头文件
block	存放块设备相关代码
crypto	存放加密、压缩、CRC 校验等算法相关代码
Documentation	存放相关说明文档，很多实用文档，包括驱动编写等
drivers	存放 Linux 内核设备驱动程序源码。驱动源码在 Linux 内核源码中占了很大比例，常见外设几乎都有可参考源码，对驱动开发而言，该目录非常重要。该目录包含众多驱动，目录按照设备类别进行分类，如 char、block、input、i2c、spi、pci、usb 等
firmware	存放处理器相关的一些特殊固件
fs	存放所有文件系统代码，如 fat、ext2、ext3、ext4、ubifs、nfs、sysfs 等
include	存放内核所需、与平台无关的头文件，与平台相关的头文件已经被移动到 arch 平台的 include 目录，如 ARM 的头文件目录<arch/arm/include/asm/>
init	包含内核初始化代码
ipc	存放进程间通信代码
kernel	包含 Linux 内核管理代码
lib	库文件代码实现
mm	存放内存管理代码
net	存放网络相关代码
samples	存放提供的一些内核编程范例
scripts	存放一些脚本文件，如 menuconfig 脚本
security	存放系统安全性相关代码
sound	存放声音、声卡相关驱动
tools	编译过程中一些主机必要工具
usr	cpio 相关实现
virt	内核虚拟机 KVM

6.5.2 内核配置

1. 内核 Makefile 文件

源码目录树顶层 Makefile 是整个内核源码管理的入口，对整个内核的源码编译起着决定性作用。编译内核时，顶层 Makefile 会按规则递归遍历内核源码的所有子目录下的 Makefile 文件，完成各子目录下内核模块的编译。

在内核源码的子目录中，几乎每个子目录都有相应的 Makefile 文件，管理着对应目录下的代码，对该目录的文件或者子目录的编译进行控制。Makefile 中有两种表示方式来控制某个文件是否编译，一种是默认选择编译，用 obj-y 表示，如：

```
obj-y += usb-host.o #默认编译 usb-host.c 文件
```

```
obj-y += gpio/ #默认编译 gpio 目录
```

另一种表示则与内核配置选项相关联，编译与否以及编译方式取决于内核配置，例如：

```
obj-$(CONFIG_WDT) += wdt.o # wdt.c 编译控制
```

```
obj-$(CONFIG_PCI) += pci/ # pci 目录编译控制
```

是否编译 wdt.c 文件，或者以何种方式编译，取决于内核配置后的变量 CONFIG_WDT 值：如果在配置中设置为[*]，则静态编译到内核，如果配置为[M]，则编译为 wdt.ko 模块，否则不编译。

2.Kconfig 文件

内核源码树每个目录下都还包含一个 Kconfig 文件，用于描述所在目录源代码相关的内核配置菜单，各个目录的 Kconfig 文件构成了一个分布式的内核配置数据库。通过 make menuconfig（make xconfig 或者 make gconfig）命令配置内核的时候，从 Kconfig 文件读取菜单，配置完毕保存到文件名为.config 的内核配置文件中，供 Makefile 文件在编译内核时使用。

3.内核配置工具

用户可以使用如下命令对内核进行配置：

- 1) make config：基于文本模式的交互式配置；
- 2) make menuconfig：基于文本模式的菜单型配置；
- 3) make oldconfig：使用已有的配置文件（.config），但是会询问新增的配置选项；
- 4) make xconfig：图形化的配置（需安装图形化系统）；

其中 make menuconfig 是最为常用的内核配置方式。下面将主要介绍 make menuconfig 的使用。

在运行 make menuconfig 前，主机必须先安装 ncurses 相关的库。在 Ubuntu 下执行如下命令完成 ncurses 的安装：

```
$sudo apt-get install libncurses5-dev
```

在 Linux 内核源码顶层目录下执行脚本文件：

./build-menuconfig.sh，可进入 Linux 内核配置主界面，如下图所示：

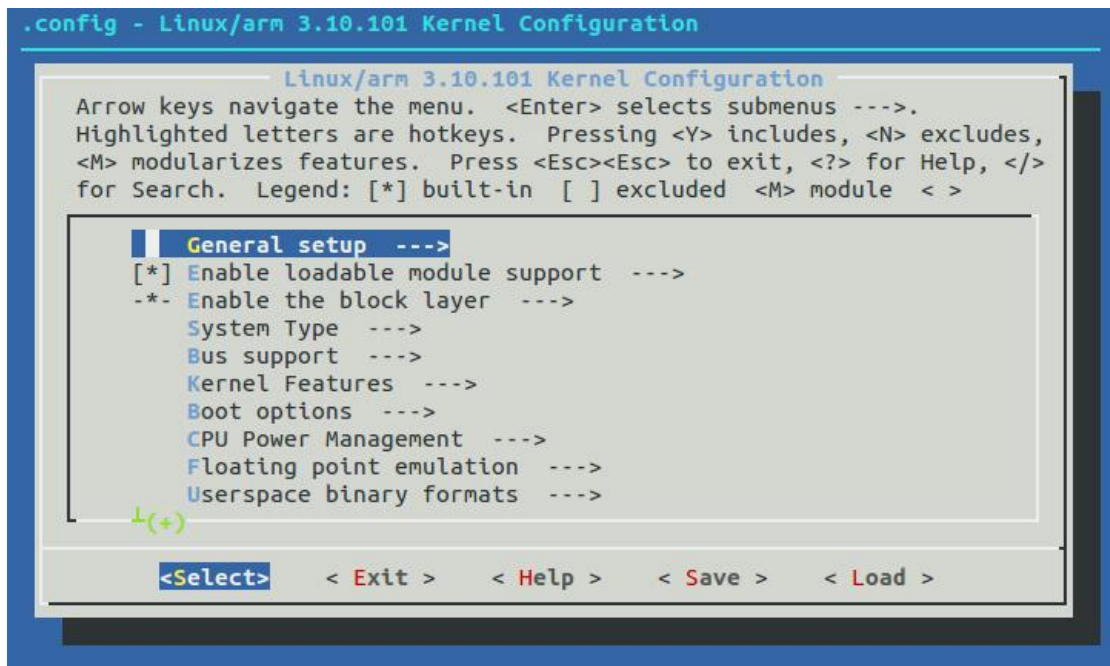


图 4.1 Linux 内核配置主界面

4.内核配置主界面由三部分组成:

- 1) 最上面部分为基本操作的简要介绍;
- 2) 中间部分为内核配置的各项菜单项;
- 3) 最下面部分为功能菜单

5.基于 Ncurses 的 Linux 内核配置界面不支持鼠标操作,必须用键盘操作。基本操作方法如下:

1) 使用上、下箭头可以选择内核配置菜单项;使用左、右箭头可以选择下排的功能菜单项;

2) 当功能菜单中的“Select”项被选中时,在某个具有子菜单的内核菜单项上按回车键可进入该菜单的子菜单;

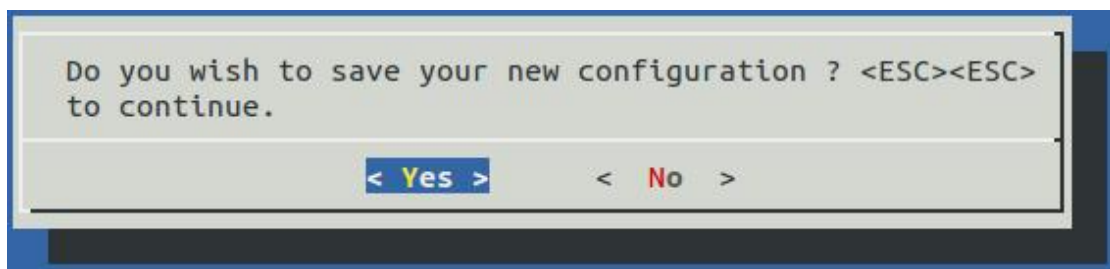
3) 对于某个菜单项,可以直接按“?”键查看该菜单项的帮助,或者用左右箭头键将功能菜单移到“Help”上,并按回车键查看帮助;

4) 根据菜单项的类型,其配置方式略有不同。可以分成三种情况,具体说明如下:

- 对于以[]开头的配置项,[*]表示选中,[]表示未选中。可以用空格键进行切换或者使用键盘快捷键(Y-选中,N-未选中)进行切换;
- 对于以<>开头的配置项,<*>表示静态编译,<M>表示编译为模块,<>表示未选中。可以使用空格键进行切换或者使用键盘快捷键(Y-静态编译,M-编译为模块,N-未选中)进行切换;
- 对于()开头的配置项,表明该配置是数值或者字符串,可以按回车键直接编辑。

5) 按斜线(/)可启用搜索功能,填入关键字后可搜索全部菜单内容;

6) 连续按两次 ESC 键或者用左右箭头键将功能菜单移到“Exit”上,并按回车键,将退回到上一级菜单。如果当前已经位于顶层菜单界面(即内核配置主界面),此时如果用户对内核配置进行了修改,则将弹出确认修改的提示画面,如下图所示:



选择“Yes”将保存对内核配置的修改并退出,选择“No”将不保存修改直接退出,连续按两次 ESC 键将重新返回内核配置画面,允许用户继续对内核配置进行修改。

内核配置完成后,其配置信息保存在内核源码顶层目录的.config 文件中。用户可以直接在命令行下将.config 文件备份为其他的文件名,以备日后使用,例如:

```
$cp .config .config_bak
```

为了使用以前备份的内核配置文件,可以直接将备份的内核配置文件覆盖内核源码顶层目录的.config 文件,例如:

```
$cp .config_bak .config
```

或者执行 make menuconfig 进入内核配置主界面,并用左右箭头选中功能菜单上的“Load”项,并按回车键,弹出配置文件加载画面,如下图所示:

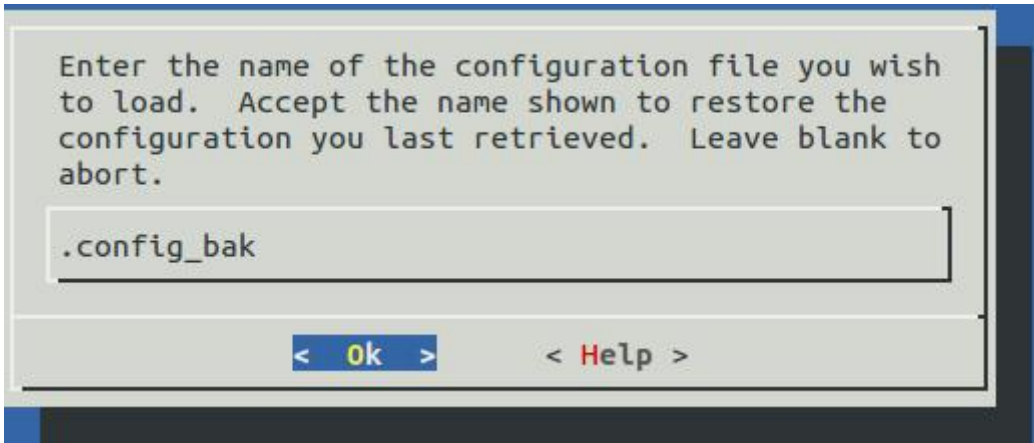


图 6.3 内核配置加载画面

在该画面中输入需要加载的备份文件的名称，选择“Ok”按钮后按回车键，此时会加载用户选择的备份文件，用户可对其进行修改并保存设置产生新的.config 文件。

6.5.3 内核编译

在“TW-STM32MP157 光盘资料\2.软件开发参考资料\3.软件源码\1.内核”目录中，包含了一个已经配置好的 Linux 内核源码文件 tw-stm32mp157_kernel_0621.tar.gz。编译 Linux 操作系统的步骤如下：

为了方便用户开发调试，提供了相对应的编译脚本文件，尽量使用脚本文件进行编译：

./build-menuconfig.sh	对内核进行配置
./build-modules.sh	编译模块 ko
./build-all.sh	编译设备树和内核

1. 将 Linux 内核源码文件 tw-stm32mp157_kernel_0621.tar.gz 拷贝到 Ubuntu 虚拟机中，此处假定将该文件拷贝到/home/tw/imx6dl 中，并对其进行解压：

```
$cd /home/twdz/MP1/kernel
$tar -xvf tw-stm32mp157_kernel_0621.tar.gz
```

2. 设置编译环境。每次编译前都需要把编译环境加载到当前 shell 来，执行如下命令：

```
$export
PATH=$PATH:/home/twdz/MP1/arm-linux/gcc-arm-9.2-2019.12-x86_64-arm-none-linux-gnueabi/bin
```

3. 清除前一次的编译结果。执行如下命令：

```
$cd /home/twdz/MP1/kernel/
$ make distclean
```

4. 使用开发板默认的配置生成编译内核时所需要的.config 文件。执行如下命令：

```
$ make stm32mp1_tw_deconfig
```

注：如果已经生成了.config 文件，或已通过 make menuconfig 对配置进行了修改，则可跳过该步骤。

5. 使用 make menuconfig 对内核进行配置。执行如下命令：

```
$ make menuconfig
```

或者执行脚本 `./build-menuconfig.sh`

注：如果不需要对内核配置进行了修改，则可跳过该步骤。

6. 编译内核或者设备树。执行如下命令或执行脚本文件：`./built-all.sh`

```
$ ./built-all.sh
```

编译完成后，会在内核源码顶层目录（即 `kernel` 目录）的 `arch/arm/boot` 子目录下生成 `uImage` 文件。

6.6 搭建 QT 编译环境

如果用户使用光盘资料里提供的虚拟机，路径为“TW-STM32MP157 光盘资料\2.软件开发参考资料\4.开发环境\ubuntu16.04”，可以直接跳过此章节。否则，需要重新搭建编译 QT 的环境。

1. 安装交叉工具链。

将光盘资料中“TW-STM32MP157 光盘资料\2.软件开发参考资料\1.编译工具”的 `st-example-image-qtwayland-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.sh` 脚本文件拷贝到虚拟机。并执行以下命令安装。

```
twdz@ubuntu:~$ chmod 777 st-example-image-qtwayland-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.sh
twdz@ubuntu:~$ ./st-example-image-qtwayland-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-snapshot.sh
```

安装过程中需要按 `y` 选择默认选项，完成后 `/opt/st` 路径下会出现 `stm32mp1` 文件夹。

2. 安装 linux QtCreator5.14.2 版本。

可以将光盘路径为“TW-STM32MP157 光盘资料\2.软件开发参考资料\1.编译工具\qt-opensource-linux-x64-5.14.2.run”安装包拷贝到虚拟机中。也可以通过命令来下载安装包，执行指令：

```
wget -c http://download.qt.io/archive/qt/5.14/5.14.2/qt-opensource-linux-x64-5.14.2.run
```

```
Connecting to mirrors.ustc.edu.cn (mirrors.ustc.edu.cn)|2001:da8:d800:95::110|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1335706944 (1.2G) [application/x-makeself]
Saving to: 'qt-opensource-linux-x64-5.14.2.run'

qt-opensource-linux 100%[=====>] 1.24G 74.6MB/s in 27s

2021-08-11 09:25:23 (47.7 MB/s) - 'qt-opensource-linux-x64-5.14.2.run' saved [1335706944/1335706944]
```

3. 修改安装包权限，执行指令：`sudo chmod 777 qt-opensource-linux-x64-5.14.2.run`

4. 执行指令：`sudo ./qt-opensource-linux-x64-5.14.2.run` 后，会弹出 QT5.14.2 的安装界面。

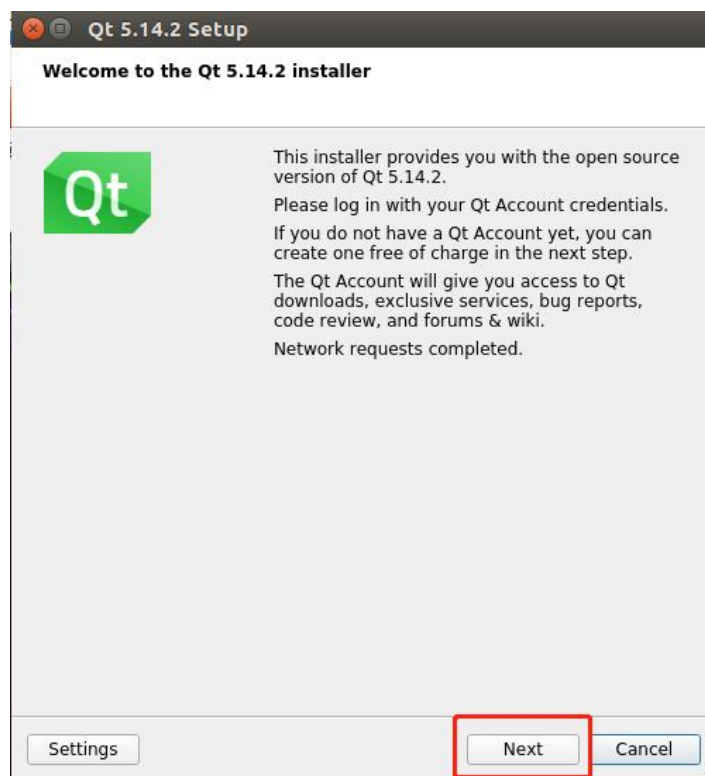


图 6.4 QtCreator 安装界面

需要输入账号和密码，如果没有，则去 Qt 官网 <https://www.qt.io/>注册一个账号。

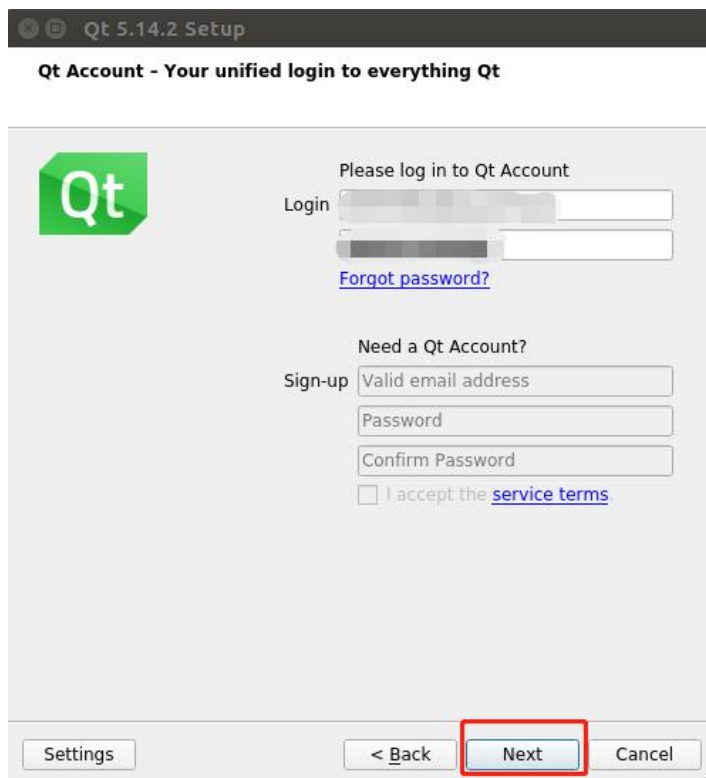


图 6.5 输入用户密码界面

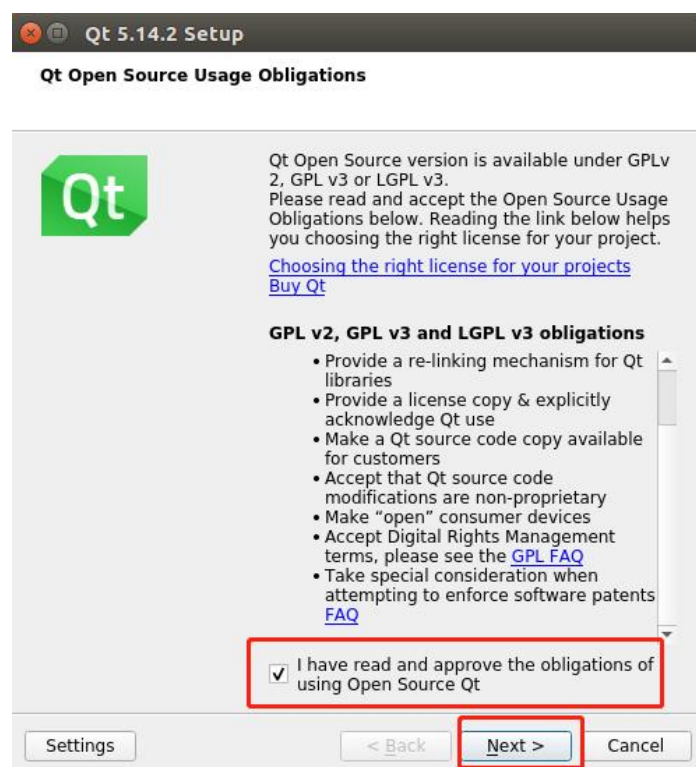


图 6.6 同意使用条款

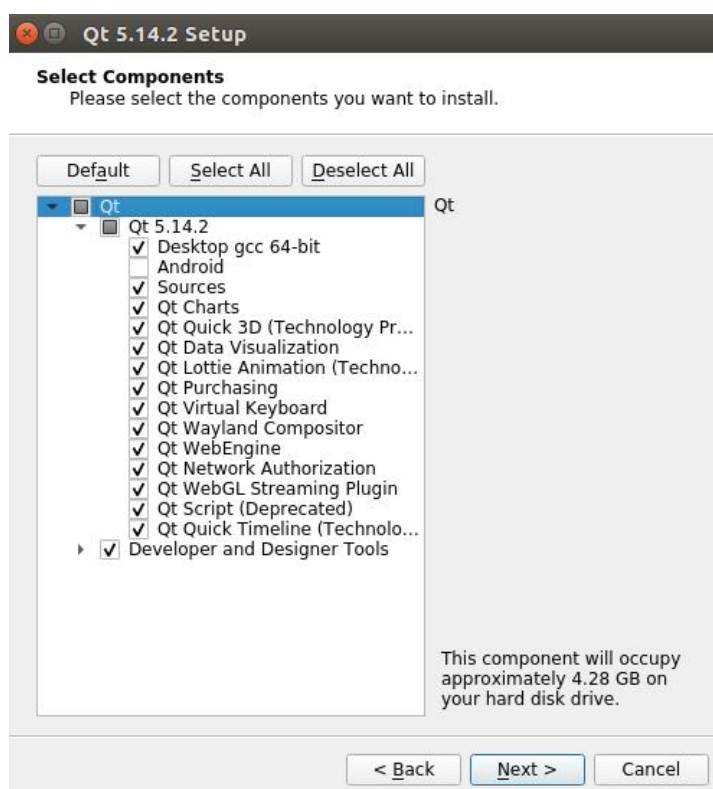


图 6.7 选择安装的选项

5. 安装完成后，打开 QtCreator 软件(按 window+F 键)。

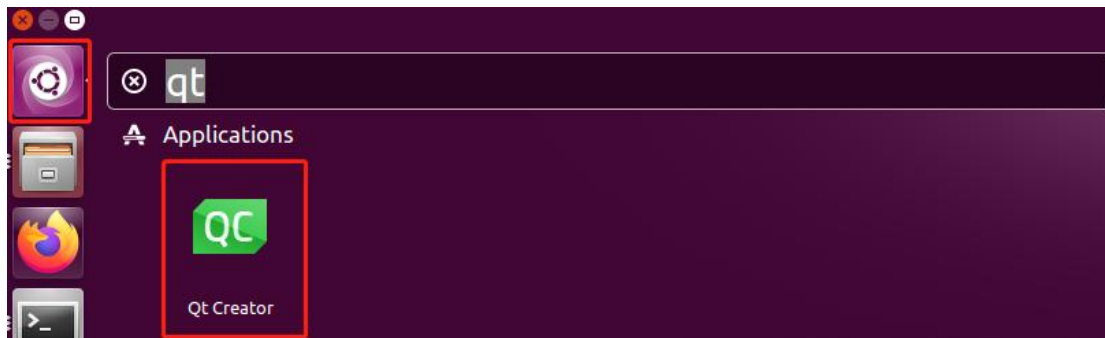


图 6.8 打开 Qt Creator 软件

6. 配置 Qt 版本的 ARM 交叉编译。Tools->Options

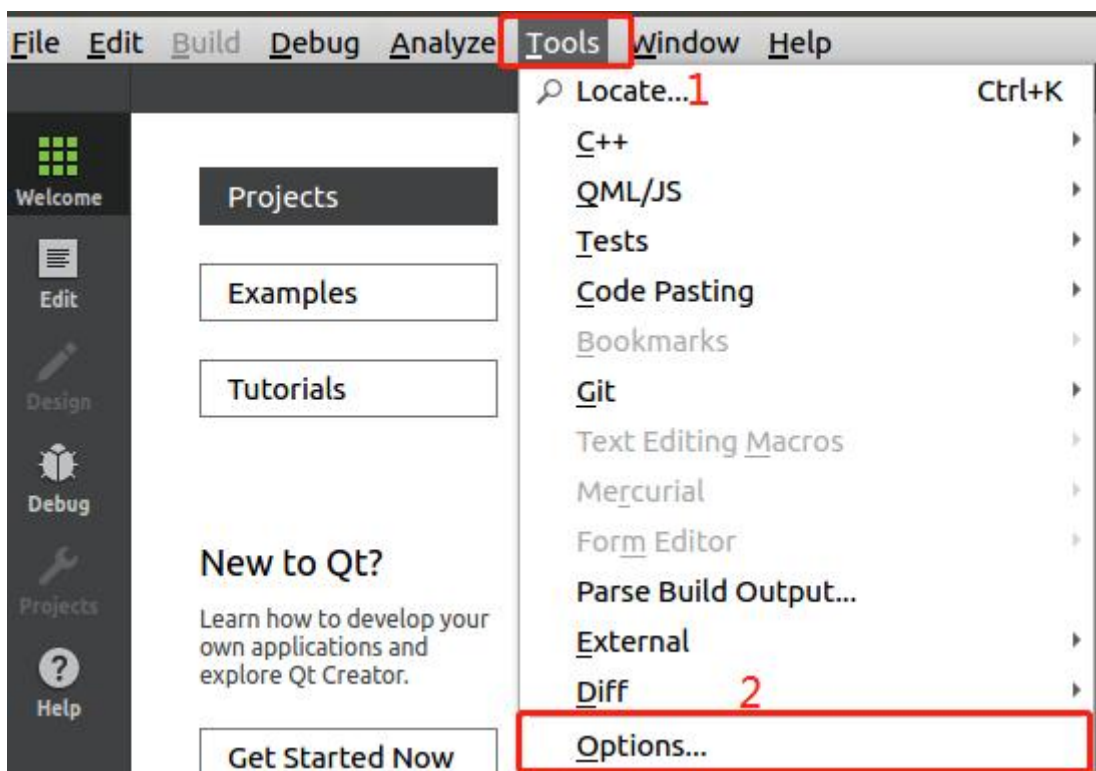


图 6.9 打开配置选项

Qt5.14.2 的交叉编译工具链 qmake，路径在
[/opt/st/stm32mp1/3.1-snapshot/sysroots/x86_64-ostl_sdk-linux/usr/bin/qmake](#)

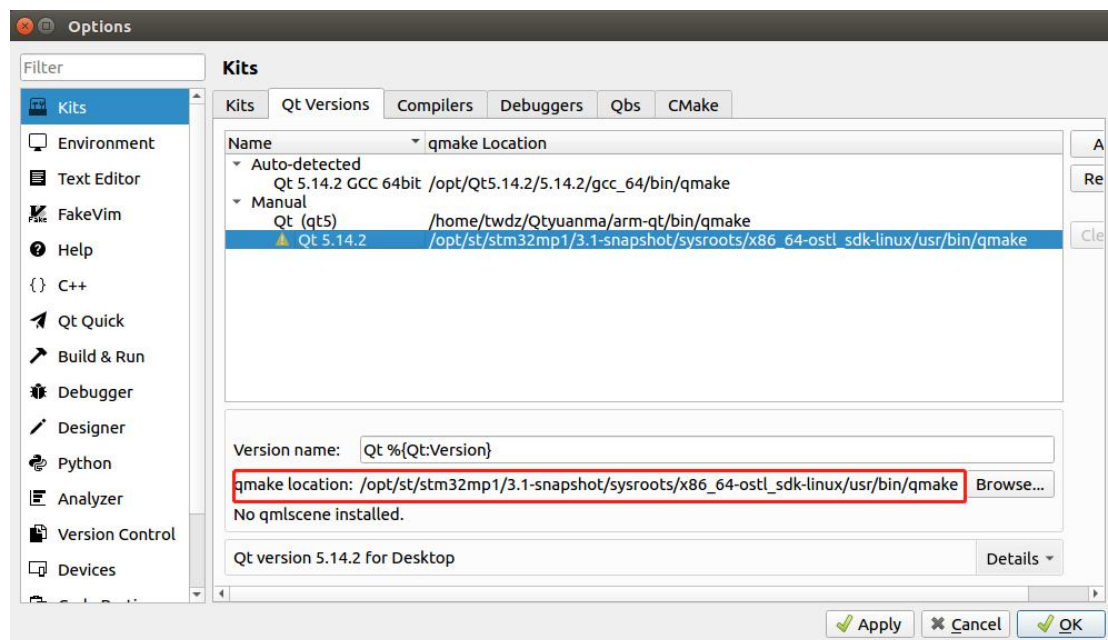


图 6.10 配置 qmake

G++交叉工具链路径为:

[/opt/st/stm32mp1/3.1-snapshot/sysroots/x86_64-ostl_sdk-linux/usr/bin/arm-ostl-linux-gnueabi/arm-ostl-linux-gnueabi-g++](#)

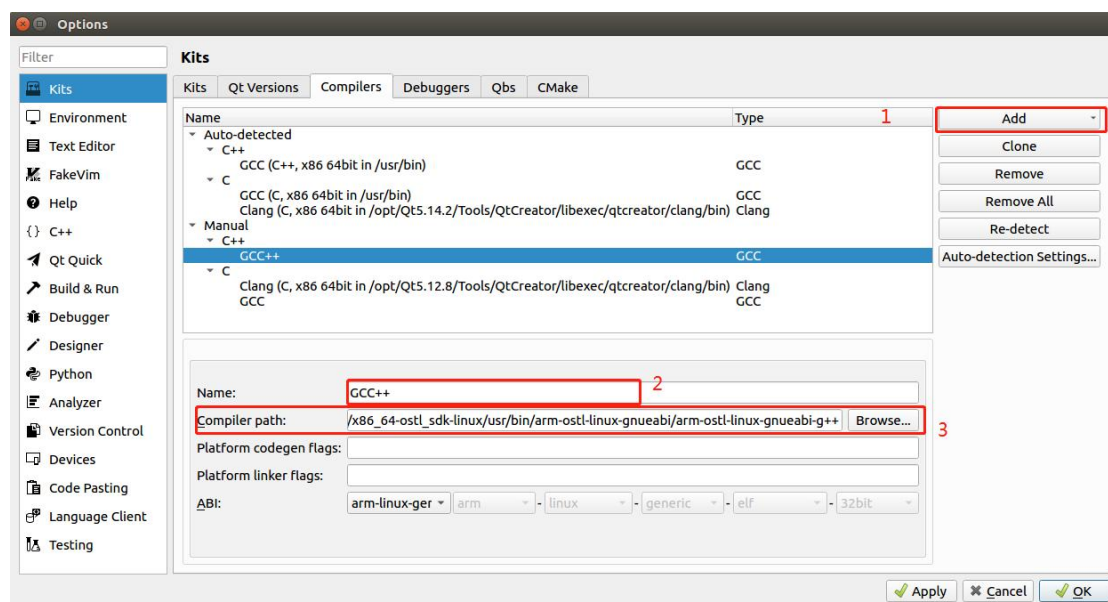


图 6.11 配置交叉编译工具链 G++

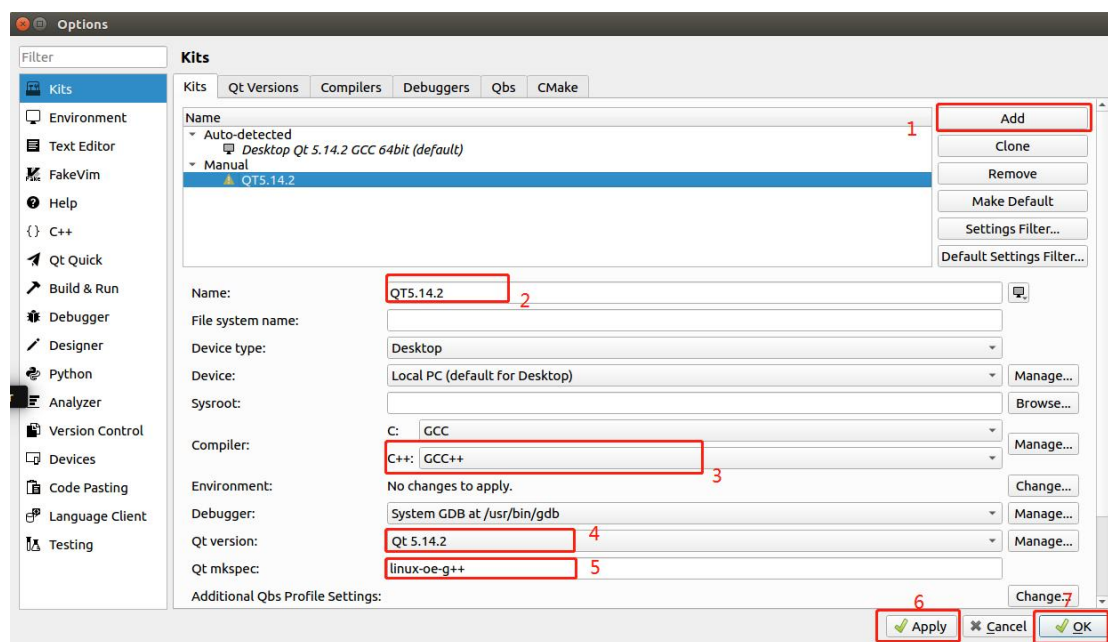


图 6.12 配置 ARM 板 Kits

6.7 编译 QT 的程序

Ubuntu 虚拟机提供了 5.14.2 版本的 QT creator，交叉工具 qmake 安装在路径 /opt/st/stm32mp1/3.1-snapshot/sysroots/x86_64-ostl_sdk-linux/usr/bin/qmake。

1. 打开 Qt Creator 应用软件，创建新的 Qt 程序，工程名为 helloworld，点击 next。

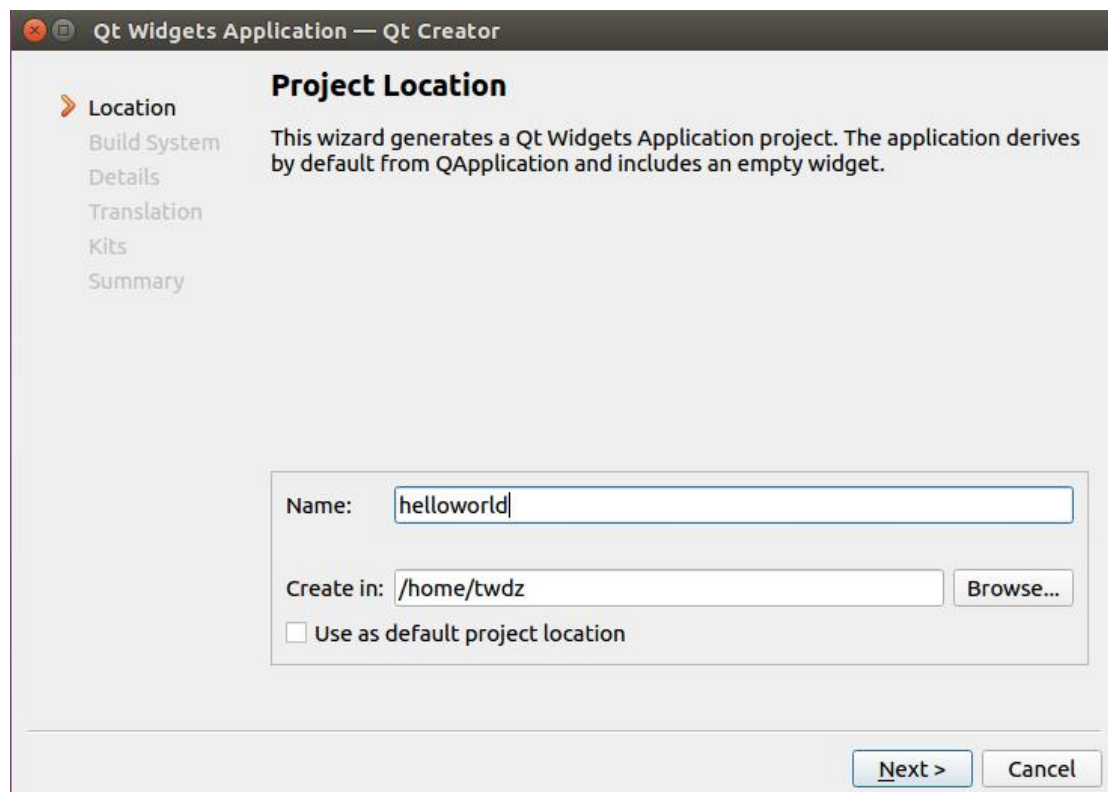


图 6.13 新建文件名

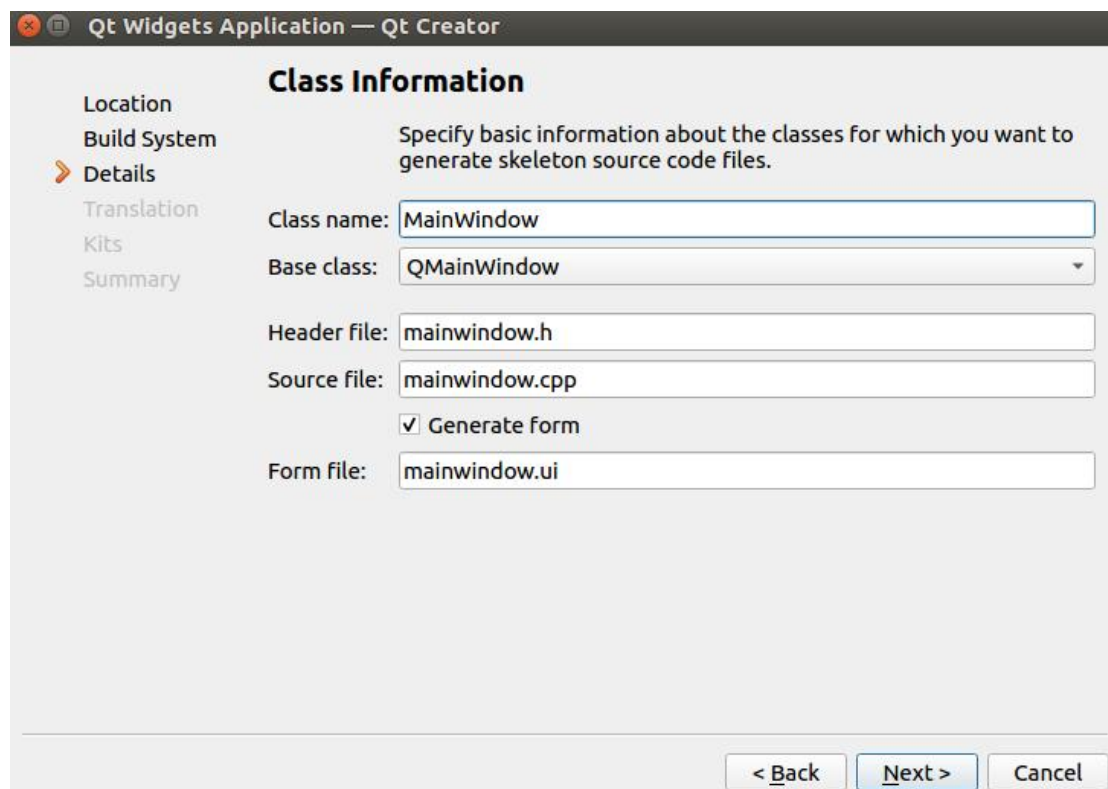


图 6.14 新建类名称

2. 选择编译工具，Desktop Qt 5.14.2 GCC 64bit 是 linux 下的编译器，QT5.14.2 是 ARM 版本的交叉工具。

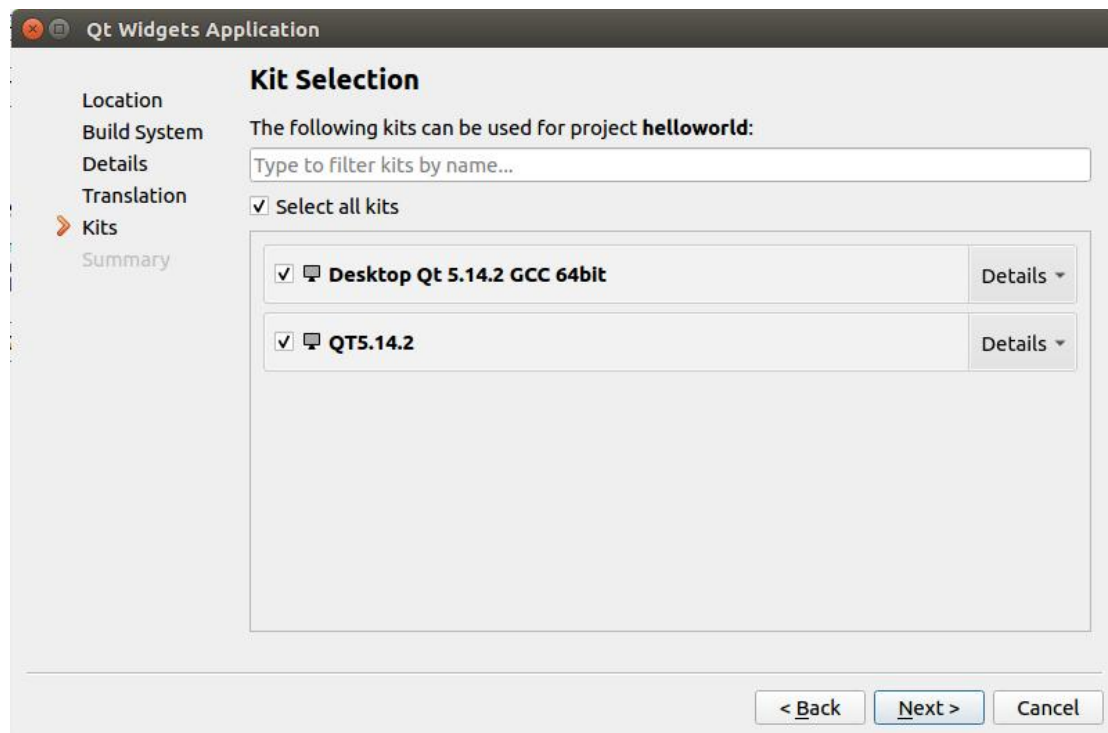


图 6.15 选择 Kit

3. 选择 ARM 版本的编译方式。

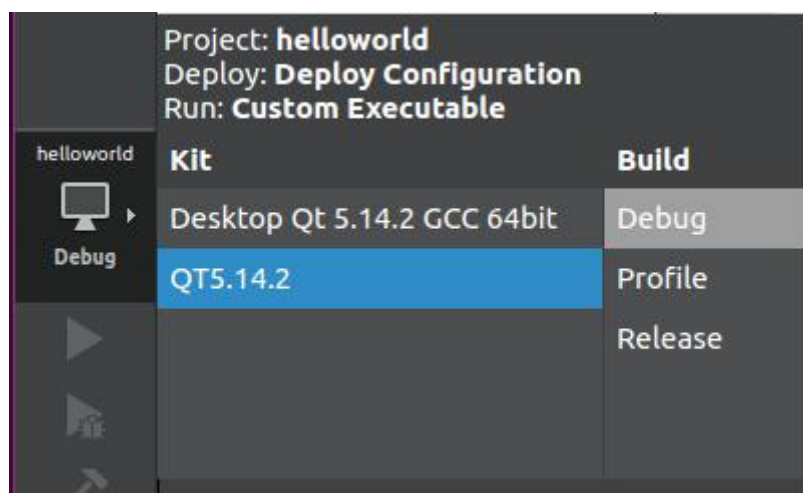
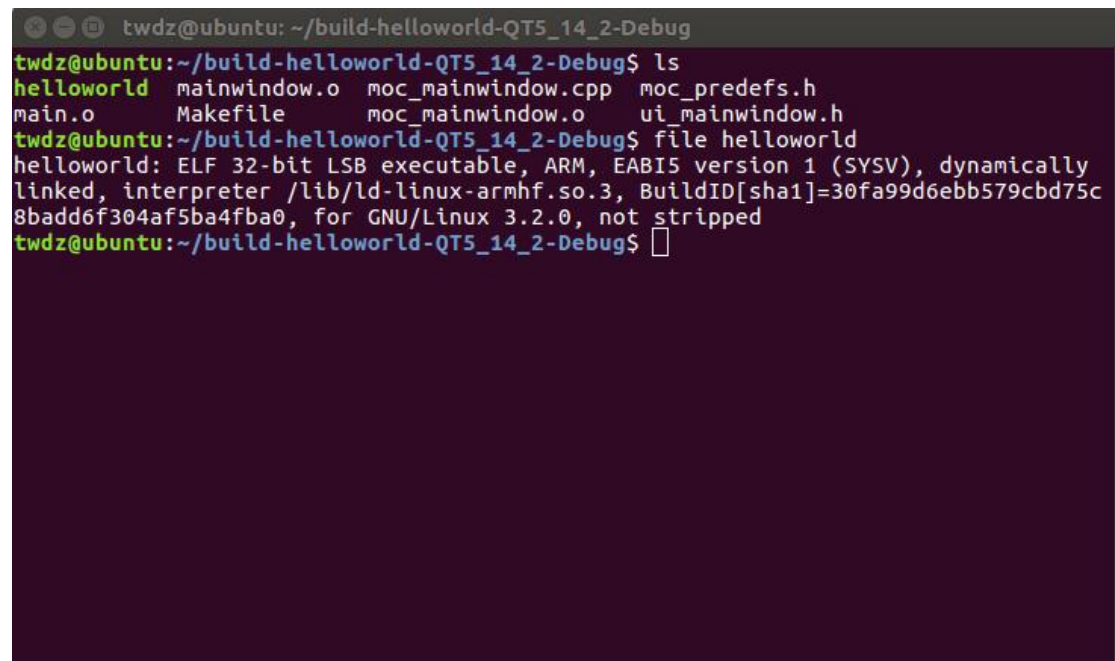


图 6.16 选择对应的 kit 编译

4. 如果编译成功，可执行文件 helloworld 在 /home/twdz/build-helloworld-QT5.14.2-Debug 路径下，通过命令：file helloworld 可以查看文件信息。



```
twdz@ubuntu: ~/build-helloworld-QT5_14_2-Debug
twdz@ubuntu:~/build-helloworld-QT5_14_2-Debug$ ls
helloworld  mainwindow.o  moc_mainwindow.cpp  moc_predefs.h
main.o      Makefile      moc_mainwindow.o    ui_mainwindow.h
twdz@ubuntu:~/build-helloworld-QT5_14_2-Debug$ file helloworld
helloworld: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically
linked, interpreter /lib/ld-linux-armhf.so.3, BuildID[sha1]=30fa99d6ebb579cbd75c
8badd6f304af5ba4fba0, for GNU/Linux 3.2.0, not stripped
twdz@ubuntu:~/build-helloworld-QT5_14_2-Debug$
```

图 6.17 查看可执行文件

5. 将 helloworld 可执行文件放到开发板上运行。

7. 免责声明

本文档提供有关广州眺望电子科技有限公司产品的信息。本文档并未授予任何知识产权的许可，并未以明示或暗示，或以禁止发言或其它方式授予任何知识产权许可。除广州眺望电子科技在其产品的销售条款和条件中声明的责任之外，概不承担任何其它责任。并且，产品的销售和 / 或使用不作任何明示或暗示的担保，包括对产品的特定用途适用性、适销性或对任何专利权、版权或其它知识产权的侵权责任等，均不作担保。广州眺望电子科技产品并非设计用于医疗、救生或维生等用途。广州眺望电子科技可能随时对产品规格及产品描述做出修改，恕不另行通知。

文档所属产品可能包含某些设计缺陷或错误，一经发现将收入勘误表，并因此可能导致产品与已出版的规格有所差异。如客户索取，可提供最新的勘误表。在订购产品之前，请您与我司销售处或分销商联系，以获取最新的规格说明。本文档中提及的含有订购号的文档以及其它文献可通过访问 <http://www.iot-tw.com/> 获得。

广州眺望电子科技有限公司保留所有权利。