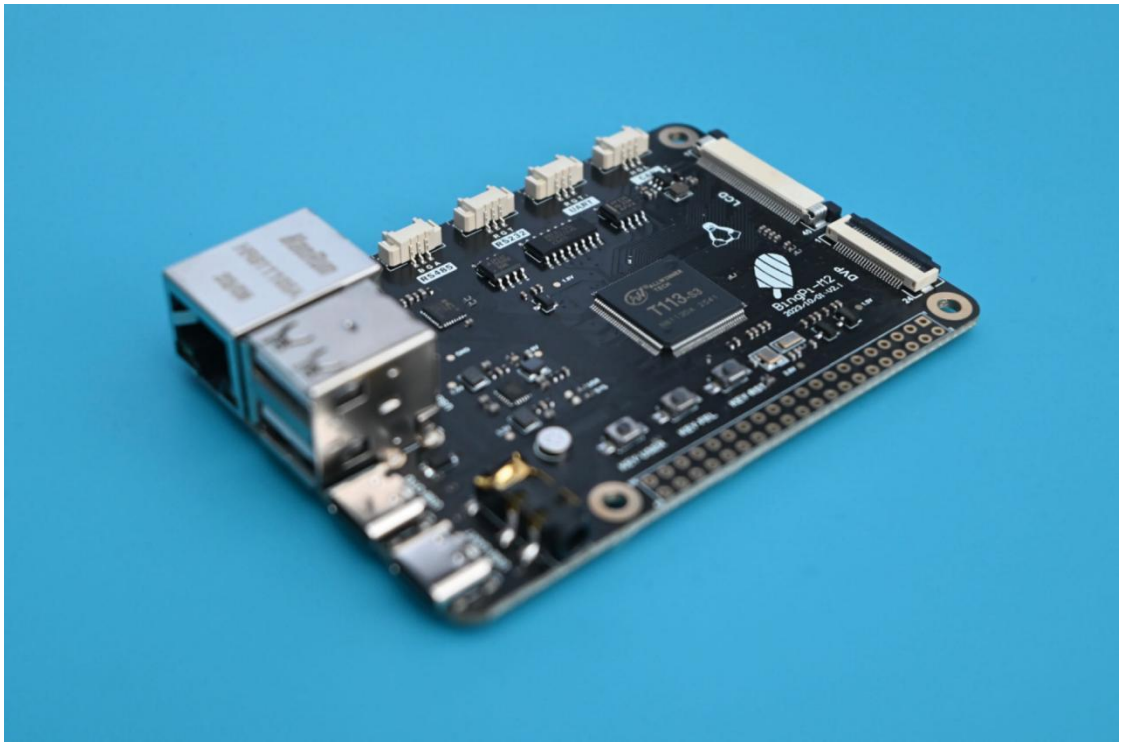
Tina Linux 参考文档

目录



版本：V1.0

日期：24/03/23

BingPi-M2 快速上手

一、Tina Linux SDK 介绍

1、SDK 结构

Tina Linux SDK 主要由构建系统、配置工具、工具链、host 工具包、目标设备应用程序、文档、脚本、linux 内核、bootloader 部分组成，下面是 Tina 主目录包含的文件和目录

```
Tina-Linux/
├── build
├── config
├── Config.in
├── device
├── dl
├── lichee
├── logs
├── Makefile
├── out
├── package
├── prebuilt
├── README.md
├── rules.mk
├── scripts
├── target
├── tmp
├── toolchain
└── tools
```

2、build 目录

build 目录存放 Tina Linux 的构建系统文件，此目录结构下主要是一系列基于 Makefile 规格编写的.mk 文件，主要的功能有：

（1）检测当前的编译环境是否满足 Tina Linux 的构建需求；

（2）生成 host 包（PC 端软件包）编译规则；

（3）生成工具链的编译规则；

（4）生成 target 包的编译规则；

（5）生成 linux kernel 的编译规则；

（6）生成系统固件的生成规则。

```
build/
├── autotools.mk
├── aw-upgrade.mk
├── board.mk
├── cmake.mk
├── config.mk
├── debug.mk
......
```

3、config 目录

config 目录主要存放 Tina Linux 中配置菜单的界面以及一些固定的配置项，该配置菜单基于内核的 mconf 规格编写。

```
config/
├──── Config-build.in
├──── Config-devel.in
├──── Config-images.in
├──── Config-kernel.in
├──── Config-systeminit.in
└──── top_config.in
```

4、device 目录

devices 目录用于存放方案的配置文件，包括内核配置、env 配置、分区表配置、sys_config.fex（全志定制板级配置文件）、board.dts（linux 标准设备树文件）等

```
device/
└──── config
    ├──── chips
    │    ├──── d1
    │    ├──── f133
    │    └──── t113
    └──── common
        ├──── cert
        ├──── debug
        ├──── dtb
        ├──── hdcp
        ├──── imagecfg
        ├──── partition
        ├──── sign_config
        ├──── toc
        ├──── tools
        └──── version
```

其中，config/chips/t113 存放 T113 平台相关的配置，其目录结构如下：

```
t113/
├──── bin
│    ├──── boot0_nand_sun8iw20p1.bin
│    ├──── boot0_sdcard_sun8iw20p1.bin
│    ├──── boot0_spinor_sun8iw20p1.bin
│    ├──── dsp0.bin
│    ├──── fes1_sun8iw20p1.bin
│    ├──── optee_sun8iw20p1.bin
│    ├──── sboot_sun8iw20p1.bin
│    └──── u-boot-sun8iw20p1.bin
```

```
├──── boot-resource
│    ├──── boot-resource
│    └──── boot-resource.ini
├──── configs
│    ├──── bingpi_m2
│    ├──── default
│    ├──── evb1
│    └──── mq_r
└──── tools
     └──── cardscript.fex
```

- bin 目录存放编译 boot 等 bin 文件，当 Tina SDK 构建或重新编译 boot 时，对应的文件会被替换。
  快捷跳转命令： cbin。
- boot-resource 目录存放开机动画等资源。
- configs 目录存放该 CHIP 平台对应的多个硬件方案配置文件。其中，default 为公共配置，bingpi-m2 对应硬件 bingpi-m2 板的方案配置，若存在更多个硬件方案，便会在该目录下新建对应的方案目录。 若公共配置目录 default 和方案配置目录中，存在相同的配置文件时，优先使用方案配置。
  快捷跳转命令：cconfigs（该命令会跳转到该目录下 linux 目录）。

以 bingpi-m2 方案为例，简述方案配置目录下，具体内容：

```
t113/configs/bingpi-m2
├──── BoardConfig.mk
├──── board.dts -> linux-5.4/board.dts
├──── boot-resource
│    └──── bootlogo.bmp
├──── env.cfg
├──── linux -> linux-5.4
├──── linux-5.4
│    ├──── board.dts
│    ├──── config-5.4
│    ├──── config-5.4-bak
│    └──── config-5.4-mem-optimize
├──── sys_config.fex
├──── sys_partition.fex
```

BoardConfig.mk 为方案板 uboot，内核选择的默认配置文件
board.dts 板级（内核）dts 配置文件，uboot-board.dts 为 uboot 设备树文件。
env.cfg 环境变量配置文件，Uboot 将此环境变量传递给内核。
linux/config-5.4 Linux5.4 内核配置文件，配置方案下默认 linux 内核功能。
sys_config.fex 打包阶段根据 sys_config 配置更新 boot0, uboot, optee 等 bin 文件的头部等信息，例如更新 dram 参数、uart 参数等。
sys_partition.fex 分区配置文件。

5、lichee 目录

lichee 目录主要存放 bootloader、linux 内核、DSP 等代码，其中 DSP 代码及编译环境因涉及 DSP 供应商科声讯版权，需单独申请。lichee 目录下结构如下：

```
Tina-Linux
├──── brandy-2.0
│    ├──── build.sh
│    ├──── spl-pub
│    ├──── tools
│    └──── u-boot-2018
└──── linux-5.4
```

6、package 目录

package 目录存放 Tina 系统支持的软件包源码和编译规则，目录按照目标软件包的功能进行分类，该目录包含了 Tina 系统全平台（包括全志 R/H/F/V/T 系列）的软件包，但是并不是所有软件包都适配了 T113 方案，部分软件包需要开发者自行适配。

```
package/
├──── add-rootfs-demo
├──── admin
├──── allwinner
├──── avs
├──── base-files
├──── busybox-init-base-files
├──── devel
├──── firmware
├──── gui
├──── kernel
├──── lang
├──── libs
├──── luci
├──── luci.mk
├──── Makefile
├──── multimedia
├──── network
├──── nn
├──── ros
├──── routing
├──── security
├──── system
├──── testtools
├──── testunit
├──── thirdparty
└──── utils
```

7、prebuild 目录

prebuild 目录存放预编译交叉编译器，目录结构如下。gcc/linux-x86/arm/toolchain-sunxi-musl/toolchain/bin/arm-openwrt-linux-gcc 即为编译 T113 所用的工具链目录

```
prebuilt/
└──── gcc
    └──── linux-x86
        ├──── aarch64
        │    ├──── aarch64-toolchain.txt
        │    ├──── toolchain-sunxi-glibc
        │    ├──── toolchain-sunxi-glibc-gcc-830
        │    ├──── toolchain-sunxi-musl
        │    └──── toolchain-sunxi-musl-gcc-830
        ├──── arm
        │    ├──── arm-toolchain.txt
        │    ├──── toolchain-sunxi-arm9-glibc
        │    ├──── toolchain-sunxi-arm9-musl
        │    ├──── toolchain-sunxi-glibc
        │    ├──── toolchain-sunxi-glibc-gcc-531
        │    └──── toolchain-sunxi-musl
        ├──── host
        │    └──── host-toolchain.txt
        └──── riscv
            ├──── riscv-toolchain.txt
            └──── toolchain-thead-glibc
```

8、scripts 目录

scripts 目录用于存放 host 端(PC 端，下同)或 target 端（小机端，即目标机器，下同）使用的一些脚本。

```
scripts/
├──── add_initramfs.sh
├──── arm-magic.sh
├──── brcmImage.pl
├──── build_rootfs.sh
├──── bundle-libraries.sh
......
```

9、target 目录

target 目录用于存放目标板相关的配置以及 sdk 和 toolchain 生成的规格。

```
target/
├── allwinner
├── Config.in
├── imagebuilder
├── Makefile
├── sdk
└── toolchain
```

10、toolchain 目录

toolchain 目录包含交叉工具链构建配置、规则。

```
toolchain/
├── binutils
├── Config.in
├── fortify-headers
├── gcc
├── gdb
├── glibc
├── info.mk
├── insight
├── kernel-headers
├── Makefile
├── musl
└── wrapper
```

11、tools 目录

tools 目录用于存放 host 端工具的编译规则。

12、out 目录

out 目录用于保存编译相关的临时文件和最终镜像文件 ，编译后自动生成此目录，以编译 bingpi-m2 方案为例说明。

```
out/
├── f133-mq_r
├── host
├── serversocket
├── t113-bingpi_m2
└── t113-mq_r
```

其中，host 目录用于存放 host 端的工具以及一些开发相关的文件。

t113-bingpi-m2 目录为方案对应的目录。方案目录下的结构如下：

```
out/t113-bingpi-m2
├── boot0_nand_sun8iw20p1.bin
├── boot0_sdcard_sun8iw20p1.bin
├── boot0_spinor_sun8iw20p1.bin
├── boot.img
├── compile_dir
├── fes1_sun8iw20p1.bin
├── image
├── md5sums
├── packages
├── rootfs.img
├── sboot_sun8iw20p1.bin
├── sha256sums
```

```
├── staging_dir
├── t113-bingpi_m2-boot.img
├── t113-bingpi_m2-uImage
├── t113-bingpi_m2-zImage
└── tina_t113-bingpi_m2_uart3.img
```

其中，

- tina_t113-bingpi_m2_uart3.img 为最终固件包(系统镜像)，串口信息通过串口输出。若使用 pack -d，则生成的固件包为 xxx_card0.img，串口信息转递到 tf 卡座输出。
- rootfs.img 为最终烧写到系统 rootfs 分区的数据，该分区默认为 squashfs 格式。
- t113-bingpi_m2-boot.img 为内核的 Image 格式镜像，用于进一步生成 uImage。
- t113-bingpi_m2-uImage 为内核的 uImage 格式镜像，若配置为 uImage 格式，则会拷贝成 boot.img。
- t113-bingpi_m2-boot.img 为内核的 boot.img 格式镜像，若配置为 boot.img 格式，则会拷贝成 boot.img
- compile_dir 为 sdk 编译 host，target 和 toolchain 的临时文件目录，存有各个软件包的源码。
- staging_dir 为 sdk 编译过程中保存各个目录结果的目录。
- packages 目录保存的是最终生成的 ipk 软件包

二、SDK 编译

1、获取环境变量

在 Tina-Linux 目录下，打开终端。

```
source build/envsetup.sh
```

2、选择硬件方案

```
lunch
```

这里我们选择 6，即 t113_bingpi_m2-tina。

3、编译

```
make
```

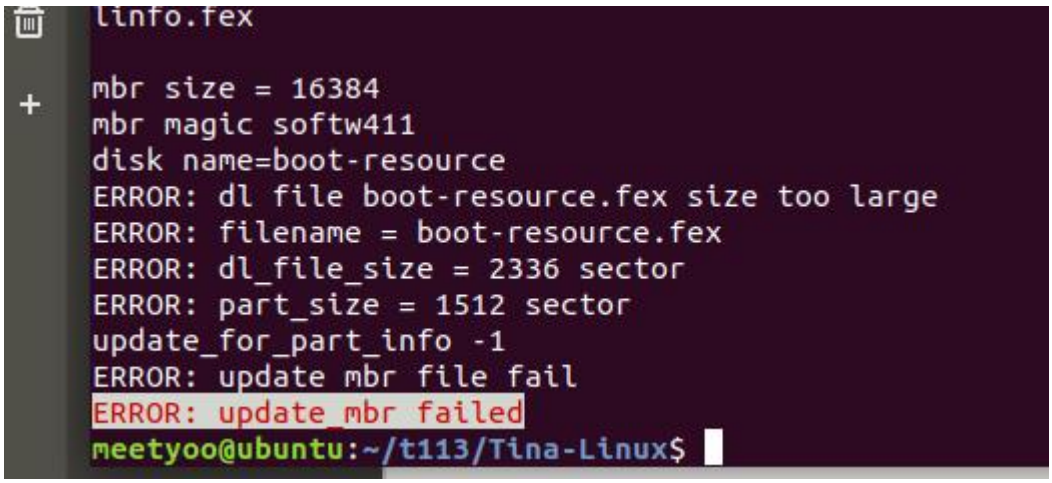
```
mboot
```

4、打包

```
pack
```

生成的 img 格式的固件在 out/t113-bingpi-m2/tina_t113-bingpi_m2_uart3.img。

常见错误：



此报错说明分区设置值小了，需要修改分区配置文件调大一些，一般根据提示修改即可，将小值改大一些，此处将 1512 改为 2336。(具体根据自己实际提示修改)

分区配置文件所在位置：

`Tina-Linux/device/config/chips/t113/configs/bingpi_m2/sys_partition.fex`

5、Tina 配置

【如果需要配置此项请在第 2 步后执行，否则忽略，此处为配置 tina 环境（根文件系统）】

```
make menuconfig
```

6、内核配置

【如果需要配置此项请在第 2 步后执行，否则忽略，此处为配置系统内核】

```
make kernel_menuconfig
```

三、SDK 烧写

我们这里生成的固件为 spi nand 的烧写固件，这里使用 PhoenixSuit 进行烧写。该工具可以在资料包中下载获得。请按下面步骤进行烧写（注意要**按顺序**，且每一步**正确操作**）

1、将上面编译好的.img 固件文件拷至 windows 下。

2、打开 PhoenixSuit，点击一键刷机，点击浏览，将路径指向刚才的.img 文件，所有复选框打勾。如下图所示。**【注意：此时任何按钮都不要点击！！！】**

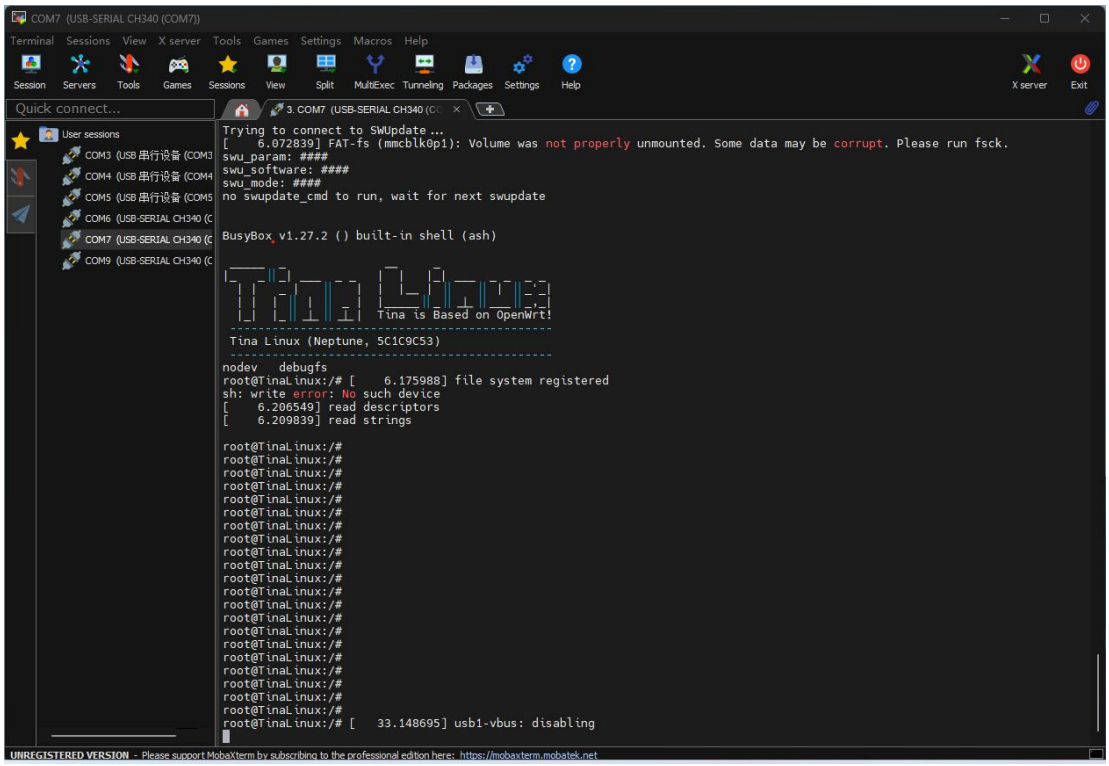
3、**先**按下 bingpi-m2 板卡上的 fel 按键，**再**将板卡的 USB OTG 接口通过 Tpye C 线缆与电脑相连，此时电脑弹出以下界面(**如果未弹出以下界面，查看电脑设备管理器里是否存在未知设备，请更新全志 usb 烧录固件驱动再试，驱动请去资料包下载获取**)。

4、点击“是”，开始烧录，直至烧录完成。

四、演示应用程序——运行 LVGL

1、将板卡的 USB UART 接口通过 Tpye C 线缆与电脑相连，查看串口端口号（如果未出现端口号，请先安装 CH340 芯片驱动，驱动可在资料包中下载）。

2、打开串口终端(波特率 115200，N/8/1)，板卡已经启动。



3、输入 lv_examples 0　回车，lvgl demo 程序即可运行。

五、演示应用程序——运行 QT

1、将板卡的 USB UART 接口通过 Tpye C 线缆与电　脑相连，查看串口端口号（如果未出现端口号，请先安装 CH340 芯片驱动，驱动可在资料包中下载）。

2、打开串口终端，板卡已经启动。

3、输入以下命令

```
cd /usr/lib/arm-qt/examples
chmod 777 animatedtiles
./animatedtiles
```

六、录音功能演示

1、查看声卡

```
ls /proc/asound/audiocodec
```

```
 Tina is Based on OpenWrt!
 ----------------------------------------------
 Tina Linux (Neptune, 5C1C9C53)
 ----------------------------------------------
nodev   debugfs
root@TinaLinux:/# [    6.061012] file system registered
sh: write error: No such device
[    6.092852] read descriptors
[    6.096101] read strings

root@TinaLinux:/#
root@TinaLinux:/# ls /proc/asound/audiocodec
id      pcm0c  pcm0p
```

2、查看声卡控制项

```
amixer -D hw:audiocodec controls
```

```
root@TinaLinux:/# amixer -D hw:audiocodec controls
numid=17,iface=MIXER,name='Headphone volume'
numid=30,iface=MIXER,name='Headphone Switch'
numid=12,iface=MIXER,name='FMINL gain volume'
numid=13,iface=MIXER,name='FMINR gain volume'
numid=2,iface=MIXER,name='ADC1 ADC2 swap'
numid=24,iface=MIXER,name='ADC1 Input FMINL Switch'
numid=25,iface=MIXER,name='ADC1 Input LINEINL Switch'
numid=23,iface=MIXER,name='ADC1 Input MIC1 Boost Switch'
numid=6,iface=MIXER,name='ADC1 volume'
numid=27,iface=MIXER,name='ADC2 Input FMINR Switch'
numid=28,iface=MIXER,name='ADC2 Input LINEINR Switch'
numid=26,iface=MIXER,name='ADC2 Input MIC2 Boost Switch'
numid=7,iface=MIXER,name='ADC2 volume'
numid=3,iface=MIXER,name='ADC3 ADC4 swap'
numid=29,iface=MIXER,name='ADC3 Input MIC3 Boost Switch'
numid=8,iface=MIXER,name='ADC3 volume'
numid=5,iface=MIXER,name='DAC volume'
numid=31,iface=MIXER,name='HpSpeaker Switch'
numid=14,iface=MIXER,name='LINEINL gain volume'
numid=15,iface=MIXER,name='LINEINR gain volume'
numid=32,iface=MIXER,name='LINEOUT Switch'
numid=16,iface=MIXER,name='LINEOUT volume'
numid=18,iface=MIXER,name='LINEOUTL Output Select'
numid=19,iface=MIXER,name='LINEOUTR Output Select'
numid=20,iface=MIXER,name='MIC1 Input Select'
numid=9,iface=MIXER,name='MIC1 gain volume'
numid=21,iface=MIXER,name='MIC2 Input Select'
numid=10,iface=MIXER,name='MIC2 gain volume'
numid=22,iface=MIXER,name='MIC3 Input Select'
numid=11,iface=MIXER,name='MIC3 gain volume'
numid=1,iface=MIXER,name='codec hub mode'
numid=4,iface=MIXER,name='digital volume'
root@TinaLinux:/#
```

3、查看音频设备

```
aplay -l
```

```
root@TinaLinux:/# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: audiocodec [audiocodec], device 0: SUNXI-CODEC 2030000.codec-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: sndspdif [sndspdif], device 0: 2036000.spdif-snd-soc-dummy-dai snd-soc-dummy-dai-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
root@TinaLinux:/#
```

4、开启音频通路

```
amixer -D hw:audiocodec cset name='MIC3 Input Select' 0 && \
amixer -D hw:audiocodec cset name='ADC3 Input MIC3 Boost Switch' 1 && \
amixer -D hw:audiocodec cset name='MIC3 gain volume' 30
```

5、录制声音

```
arecord -D hw:audiocodec -f S16_LE -t wav -r 16000 -d 5 test.wav
```

6、播放（可参考第七章音频播放功能）

七、音频播放演示功能（插入音响或耳机）

1、开启音频通路

```
amixer -D hw:audiocodec cset name='Headphone Switch' 1 && \
amixer -D hw:audiocodec cset name='Headphone volume' 25
```

2、播放音频（wav 文件）

```
aplay -D hw:audiocodec wfxy.wav
```

八、视频播放演示

1、将视频放入 U 盘或者 SD 卡中，这里我们以将视频放入 U 盘为例。

2、挂载 U 盘。（如果 u 盘未自动挂载，我们手动挂载 U 盘）

```
mount /dev/sda1 /mnt/exUDISK
```

3、播放视频

```
cd /mnt/exUDISK
tplayerdemo test.mp4
```

九、Wifi 联网测试演示

1、wifi 驱动安装

开发板中采用的是 USB 接口的 wifi 模块（RTL8188EUS），出厂镜像中已经编译好了驱动，首先我们找到驱动位置。

```
cd /lib/modules/5.4.61/
ls
```

```
root@TinaLinux:/# cd /lib/modules/5.4.61/
root@TinaLinux:/lib/modules/5.4.61# ls
gt9xxnew_ts.ko     r8188eu.ko         xt_LOG.ko          xt_mac.ko
ipt_REJECT.ko      sunxi_gpadc.ko     xt_TCPMSS.ko       xt_mark.ko
iptable_filter.ko  vin_io.ko          xt_comment.ko      xt_multiport.ko
ov5640.ko          vin_v4l2.ko        xt_limit.ko        xt_time.ko
```

按以下命令安装驱动

```
insmod r8188eu.ko
```

```
root@TinaLinux:/lib/modules/5.4.61# insmod r8188eu.ko
[  642.747135] r8188eu: module is from the staging directory, the quality is unknown, you have been warned.
[  642.761513] Chip Version Info: CHIP_8188E_Normal_Chip_TSMC_A_CUT_1T1R_RomVer(0)
[  642.803917] usbcore: registered new interface driver r8188eu
```

2、查找网卡

```
ifconfig -a
```

```
root@TinaLinux:/lib/modules/5.4.61# 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:36

eth0      Link encap:Ethernet  HWaddr A6:40:6E:44:ED:6D
          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:39

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  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)

wlan0     Link encap:Ethernet  HWaddr 00:33:05:A0:22:EE
          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、配置 wifi 参数

配置 wifi 参数这里指的就是设置我们要连接 wifi 的名字和密码

修改配置文件（/etc/wpa_supplicant.conf），填写 wifi 名字和密码

```
vi /etc/wpa_supplicant.conf
```

```
ctrl_interface=/var/lock/wpa_supplicant
ctrl_interface_group=0
ap_scan=1
network={
    ssid="[redacted]"
    scan_ssid=1
    key_mgmt=WPA-EAP WPA-PSK IEEE8021X NONE
    pairwise=TKIP CCMP
    group=CCMP TKIP WEP104 WEP40
    psk="[redacted]"
    priority=5
}
```

ssid 为要连接的 wifi 名字

psk 为要连接的 wifi 密码

4、联网

```
ifconfig wlan0 up
wpa_supplicant -D wext -c /etc/wpa_supplicant.conf -i wlan0 &
```

动态分配 ip

```
udhcpc -i wlan0
```

5、ping 外网测试

```
ping baidu.com
```

```
root@TinaLinux:/lib/modules/5.4.61# ping baidu.com
PING baidu.com (110.242.68.66): 56 data bytes
64 bytes from 110.242.68.66: seq=0 ttl=53 time=35.834 ms
64 bytes from 110.242.68.66: seq=1 ttl=53 time=26.388 ms
64 bytes from 110.242.68.66: seq=2 ttl=53 time=38.681 ms
64 bytes from 110.242.68.66: seq=3 ttl=53 time=25.754 ms
64 bytes from 110.242.68.66: seq=4 ttl=53 time=35.446 ms
64 bytes from 110.242.68.66: seq=5 ttl=53 time=23.012 ms
64 bytes from 110.242.68.66: seq=6 ttl=53 time=31.696 ms
64 bytes from 110.242.68.66: seq=7 ttl=53 time=27.541 ms
64 bytes from 110.242.68.66: seq=8 ttl=53 time=26.728 ms
64 bytes from 110.242.68.66: seq=9 ttl=53 time=29.792 ms
64 bytes from 110.242.68.66: seq=10 ttl=53 time=34.235 ms
^C
--- baidu.com ping statistics ---
11 packets transmitted, 11 packets received, 0% packet loss
round-trip min/avg/max = 23.012/30.464/38.681 ms
```

6、一些命令

```
#查询网卡状态
wpa_cli -p/tmp/lock/wpa_supplicant -iwlan0 status

#搜索附近网络功能  no/ok
wpa_cli -p/tmp/lock/wpa_supplicant -i wlan0 scan

#搜索附近网络,并列出结果
wpa_cli -p/tmp/lock/wpa_supplicant -i wlan0 scan_resul

#查看当前连接的是哪个网络
wpa_cli -p/tmp/lock/wpa_supplicant -i wlan0 list_network
```

十、CAN 通信测试演示

1、CAN 测试工具

这里我已经将 can 通信需要用到的 ip 以及 canutils 等编译好放在了虚拟机中，我们将其拷入 u 盘，从而再拷入开发板中。

其中 ip 拷入开发板根文件系统中的/sbin 目录下

canutils 下的所有内容拷入根文件系统的根目录下

```
mount /dev/sda1 /mnt/exUDISK
cd /mnt/exUDISK/
cp ip /sbin/
cp -r canutils/* /
```

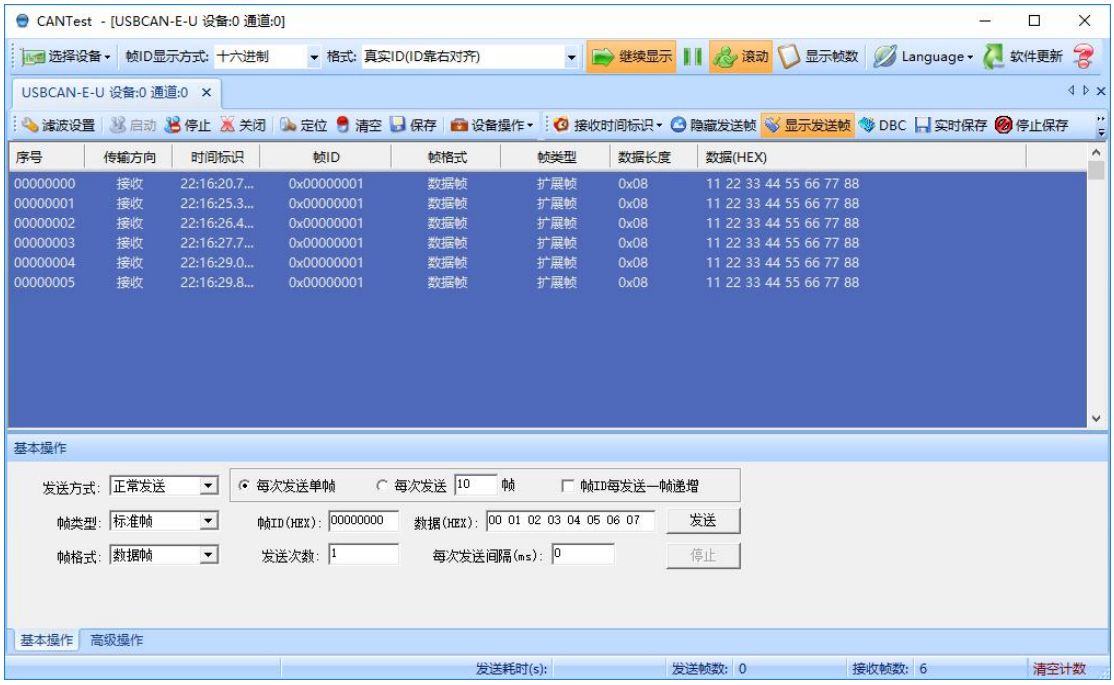
2、配置 can 波特率并启动

```
ip link set can0 up type can bitrate 500000
```

3、发送数据

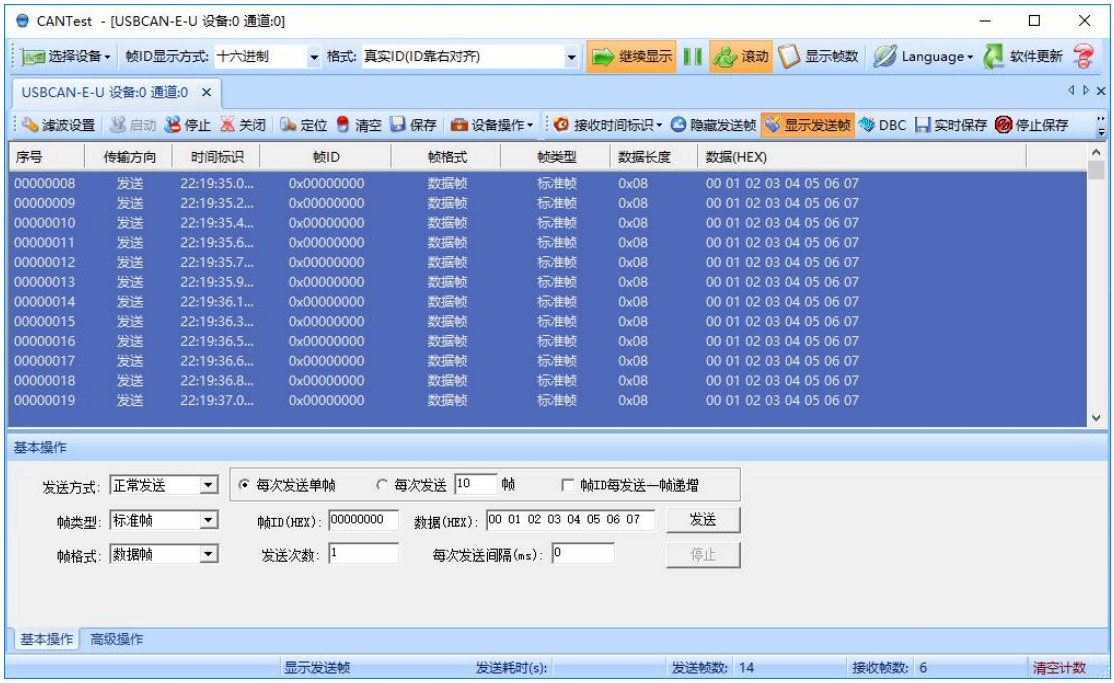
```
cansend can0 -e 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
```

```
root@TinaLinux:/# cansend can0 -e 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
interface = can0, family = 29, type = 3, proto = 1
root@TinaLinux:/# cansend can0 -e 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
interface = can0, family = 29, type = 3, proto = 1
root@TinaLinux:/# cansend can0 -e 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
interface = can0, family = 29, type = 3, proto = 1
root@TinaLinux:/#
root@TinaLinux:/# cansend can0 -e 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
interface = can0, family = 29, type = 3, proto = 1
root@TinaLinux:/# cansend can0 -e 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88
interface = can0, family = 29, type = 3, proto = 1
```



4、接收数据

```
candump can0
```




```
root@TinaLinux:/# candump can0
interface = can0, family = 29, type = 3, proto = 1
<0x000> [8] 00 01 02 03 04 05 06 07
<0x000> [8] 00 01 02 03 04 05 06 07
<0x000> [8] 00 01 02 03 04 05 06 07
<0x000> [8] 00 01 02 03 04 05 06 07
<0x000> [8] 00 01 02 03 04 05 06 07
<0x000> [8] 00 01 02 03 04 05 06 07
```

十一、以太网测试演示

1、查看网卡

将开发板通过网线与路由器相连

```
ifconfig -a
```

```
root@TinaLinux:/# 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:36

eth0      Link encap:Ethernet  HWaddr 1E:1B:11:00:D3:D8
          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:39

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  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)
```

2、启动网卡

```
ifconfig eth0 up
```

```
root@TinaLinux:/# ifconfig eth0 up
[  120.561992] libphy: 4500000.eth: probed
[  120.566613] sunxi-gmac 4500000.eth eth0: eth0: Type(7) PHY ID 02430c54 at 1 IRQ poll (4500000.eth-0:01)
root@TinaLinux:/# [  122.668496] sunxi-gmac 4500000.eth eth0: Link is Up - 100Mbps/Full - flow control off
[  122.677342] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
```

3、获取 ip

```
udhcpc
```

```
root@TinaLinux:/# udhcpc
udhcpc: started, v1.27.2
udhcpc: sending discover
udhcpc: sending select for 192.168.1.4
udhcpc: lease of 192.168.1.4 obtained, lease time 259200
udhcpc: ifconfig eth0 192.168.1.4 netmask 255.255.255.0 broadcast +
udhcpc: setting default routers: 192.168.1.1
```

4、ping 外网测试

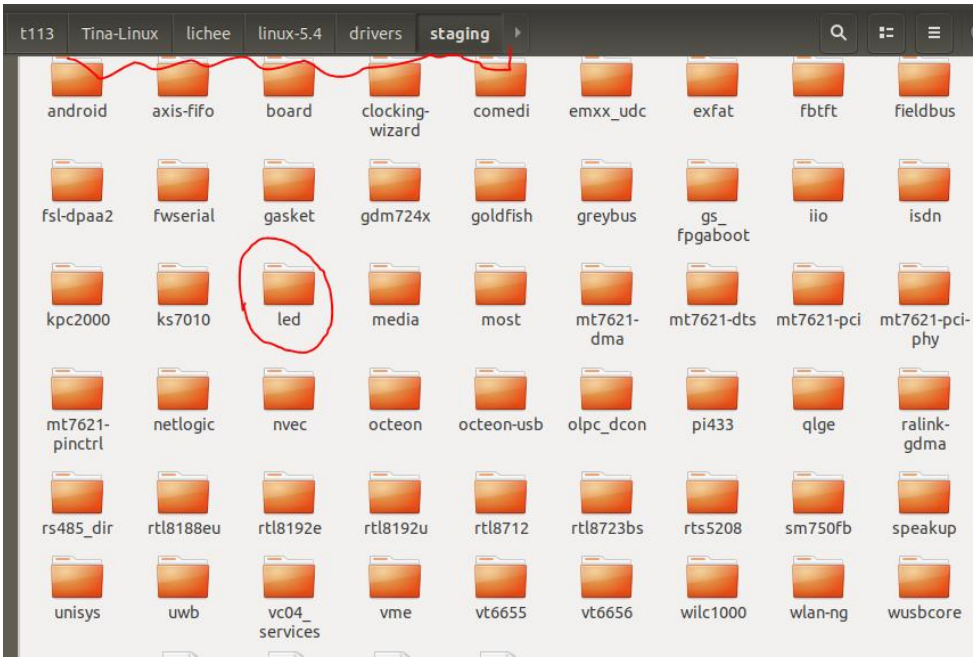
```
ping baidu.com
```

```
root@TinaLinux:/# ping baidu.com
PING baidu.com (110.242.68.66): 56 data bytes
64 bytes from 110.242.68.66: seq=0 ttl=54 time=20.797 ms
64 bytes from 110.242.68.66: seq=1 ttl=54 time=20.959 ms
64 bytes from 110.242.68.66: seq=2 ttl=54 time=20.838 ms
64 bytes from 110.242.68.66: seq=3 ttl=54 time=20.867 ms
^C
--- baidu.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 20.797/20.865/20.959 ms
```

十二、LED 测试

1、LED 驱动程序

LED 驱动程序已经写好，请到资料包中驱动开发笔记文件夹下载，下载后将 led 驱动文件夹放入 linux 内核 drivers/staging 文件夹下，如下图



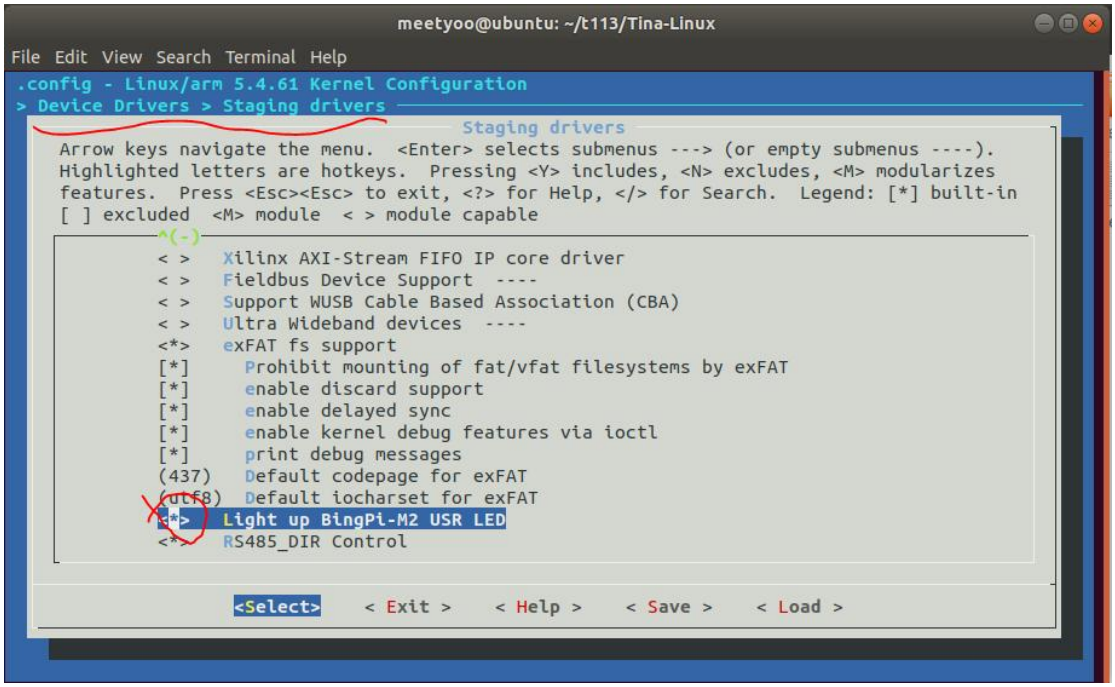
修改 drivers/staging 文件夹下的 Kconfig 文件，增加以下内容

```
source "drivers/staging/led/Kconfig"
```

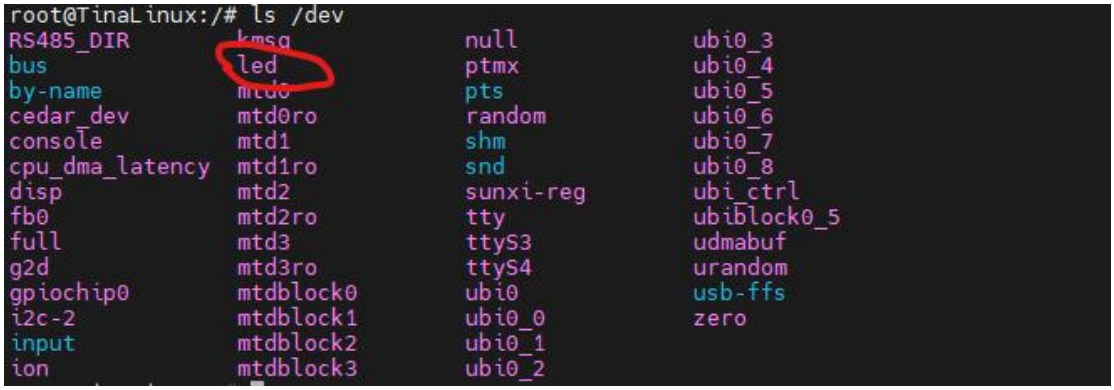
修改 drivers/staging 文件夹的 Makefile 文件，增加以下内容

```
obj-$(CONFIG_USR_LED)     += led/
```

配置内核开启 led 驱动编译，如下图



保存后，重新编译 SDK 并打包，烧录。系统启动后，可以看到/dev 下多出来个 led 的设备，此时 LED 驱动已经添加完毕。



2、LED 应用程序

LED 应用程序已经写好，请到资料包应用开发笔记文件夹下载，下载后我们将其放入虚拟机内（可自己新建一个文件夹，用于存放应用程序），这里我们不再编写 makefile 文件，打开终端输入以下命令，直接编译.c 文件产生可执行文件进行测试。

```
arm-openwrt-linux-gcc led.c -o led_flash
```

此时产生一个二进制可执行文件 led_flash

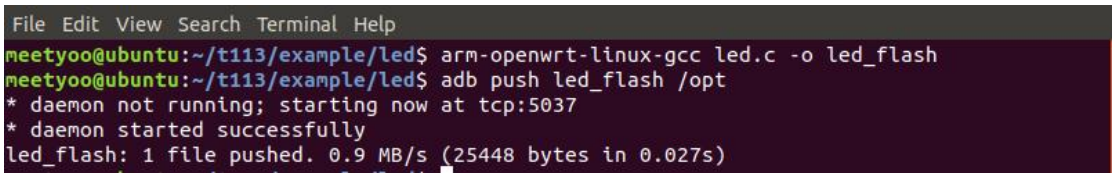
3、运行测试

我们可以通过 U 盘或者 ADB 的方式将产生的可执行文件放入开发板运行。

这里通过 ADB 方式，

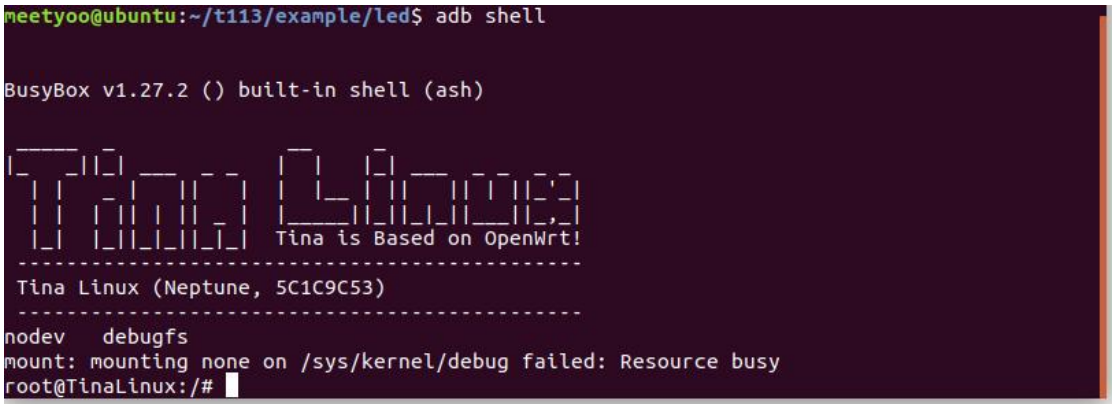
将开发板的 USB • OTG 接口通过 TYPE-C USB 线与电脑相连，电脑会识别出一个 ADB 的 USB 设备，将该设备连入虚拟机内，在终端输入以下命令，将可执行文件拷入开发板。

```
adb push led_flash /opt
```



继续输入以下命令，可打开终端。

```
adb shell
```



如果此时开发板的 led 正在闪烁，请先关掉此进程，然后执行 led_flash 的程序

```
ps
```

```
  75 root         0 SW   [irq/198-4020000]
  76 root         0 IW<  [ipv6_addrconf]
  77 root         0 SW   [ubi_bgt0d]
  78 root         0 IW<  [kworker/1:1H-kb]
  79 root         0 IW<  [kworker/0:1H-kb]
  80 root         0 IW   [kworker/0:3-eve]
 104 root         0 SW   [ubifs_bgt0_7]
 134 root         0 SW   [ubifs_bgt0_8]
 136 root         0 IW<  [goodix_wq]
 137 root         0 IW<  [goodix_wq]
 152 root       688 S    ./opt/led_flash
 158 root      1452 S    /bin/adbd -D
 161 root       696 S    /sbin/swupdate-progress -w
 181 root      1048 S    -/bin/sh
 218 root         0 IW   [kworker/u4:2-ev]
 222 root      1048 S    /bin/sh
 232 root      1048 R    ps
root@TinaLinux:/#
```

```
kill 152
cd /opt
./led_flash &
```

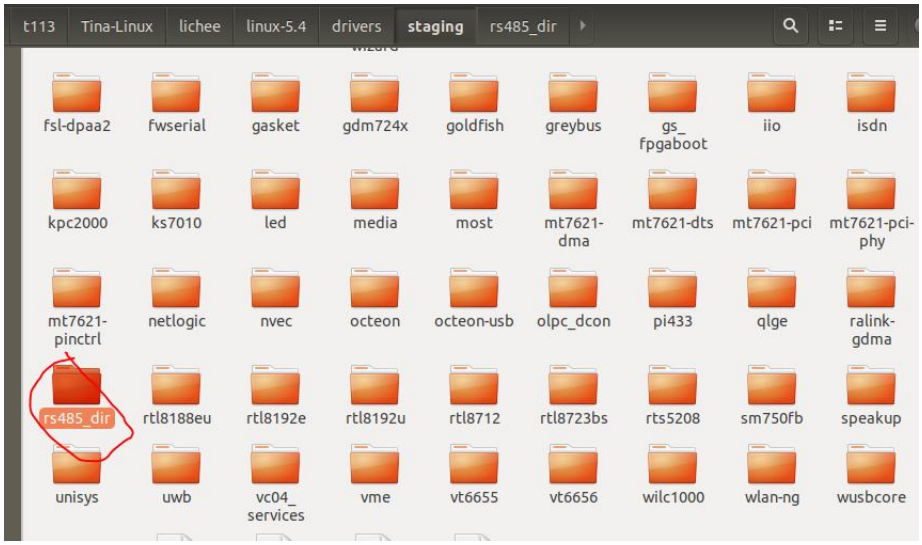
此时，led 又闪烁起来了。

十三、RS-485 测试

1、RS-485 驱动程序

BingPi-M2 开发板中有一路 RS-485 通信接口，我们知道 RS-485 属于半双工通信，也就是说接收数据的时候无法发送数据，发送数据的时候无法接收数据。RS-485 通信有一个方向控制信号，这里该信号的控制我们没有采用硬件方式，而是采用一个 GPIO 管脚来控制方向，从而控制 RS-485 的数据收发。

RS-485 驱动也即是这个方向控制 IO 的驱动，驱动程序已经写好，请到资料包中驱动开发笔记下载，下载后将 rs485_dir 驱动文件夹放入 linux 内核 drivers/staging 文件夹下，如下图。



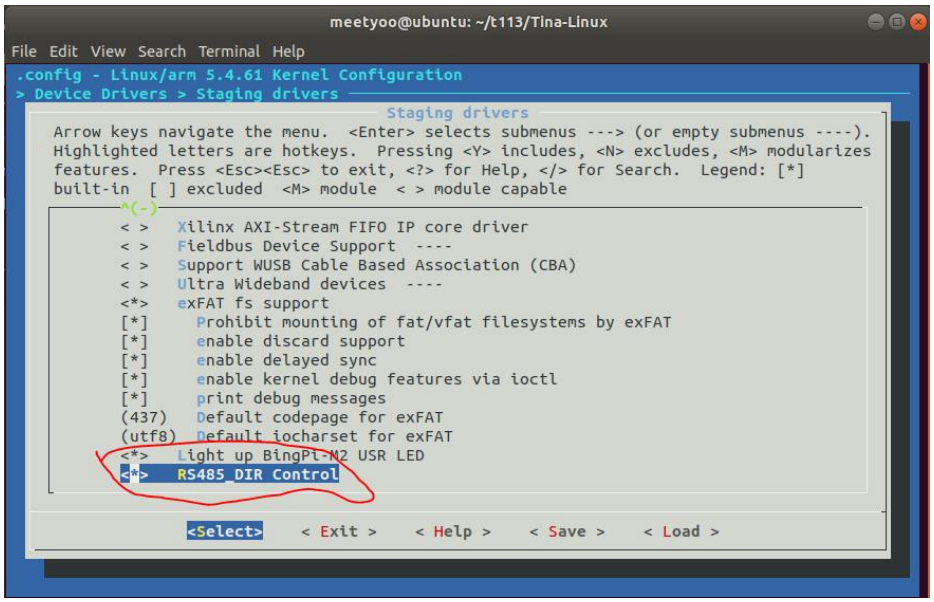
修改 drivers/staging 文件夹下的 Kconfig 文件，增加以下内容

```
source "drivers/staging/rs485_dir/Kconfig"
```

修改 drivers/staging 文件夹下的 Makefile 文件，增加以下内容

```
obj-$(CONFIG_RS485_DIR)       += rs485_dir/
```

配置内核开启 RS-485 DIR 驱动编译，如下图



保存后，重新编译 SDK 并打包，烧录。系统启动后，可以看到/dev 下多出来个 rs485_dir 的设备，此时 rs485_dir 驱动已经添加完毕，我们可以通过控制此设备从而控制 RS485 数据的收发。

```
root@TinaLinux:/# ls /dev
bus                led                ptmx               ubi0_3
by-name            mtd0               pts                ubi0_4
cedar_dev          mtd0ro             random             ubi0_5
console            mtd1               rs485_dir          ubi0_6
cpu_dma_latency    mtd1ro             shm                ubi0_7
disp               mtd2               snd                ubi0_8
fb0                mtd2ro             sunxi-reg          ubi_ctrl
full               mtd3               tty                ubiblock0_5
g2d                mtd3ro             ttyS3              udmabuf
gpiochip0          mtdblock0          ttyS4              urandom
i2c-2              mtdblock1          ubi0               usb-ffs
input              mtdblock2          ubi0_0             zero
ion                mtdblock3          ubi0_1
kmsg               null               ubi0_2
```

2、RS-485 应用程序

RS-485 的应用程序已经写好，请到资料包应用开发笔记文件夹下载，下载后我们将其放入虚拟机内（可自己新建一个文件夹，用于存放应用程序），这里我们不再编写 makefile 文件，打开终端输入以下命令，直接编译.c 文件产生可执行文件进行测试（这里的 dir 方向控制是通过发送完数据三个字符时间拉高 dir 的）。

```
arm-openwrt-linux-gcc rs485_test.c -o rs485_test
```

此时产生一个二进制可执行文件 rs485_test

3、运行测试

我们可以通过 U 盘或者 ADB 的方式将产生的可执行文件放入开发板运行。

这里通过 ADB 方式，

将开发板的 USB • OTG 接口通过 TYPE-C USB 线与电脑相连，电脑会识别出一个 ADB 的 USB 设备，将该设备连入虚拟机内，在终端输入以下命令，将可执行文件拷入开发板。

```
adb push rs485_test /opt
```

```
meetyoo@ubuntu: ~/t113/example/rs485
File  Edit  View  Search  Terminal  Help
meetyoo@ubuntu:~/t113/example/rs485$ adb push rs485_test /opt
* daemon not running; starting now at tcp:5037
* daemon started successfully
rs485_test: 1 file pushed. 0.9 MB/s (26648 bytes in 0.030s)
```

继续输入以下命令，可打开终端。

```
adb shell
```

```
meetyoo@ubuntu:~/t113/example/rs485$ adb shell


BusyBox v1.27.2 () built-in shell (ash)

 _____  _                 __      _
|_   _||_| ___  _ _    |  |    |_| ___  _ _  _ _
  | |    _ |   ||  _|  |  |__  | ||   ||  | ||_'_|
  | |  | || | || _ |  |_____||_||_|_||___||_,_|
  |_|  |_||_|_||_|_|  Tina is Based on OpenWrt!
 ---------------------------------------------
 Tina Linux (Neptune, 5C1C9C53)
 ---------------------------------------------
nodev   debugfs
mount: mounting none on /sys/kernel/debug failed: Resource busy
root@TinaLinux:/#
```

然后执行 rs485_test 程序

```
cd /opt
./rs485_test &
```

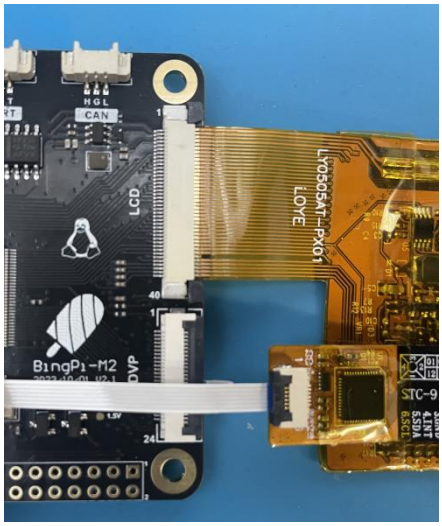
此时 RS-485 测试程序已经运行，通过 USB 转 RS-485 模块向开发板发送数据，此时开发板会返接收到的数据，说明 RS-485 测试通过。（注意：RS-485 的接线，A 接 A，B 接 B）

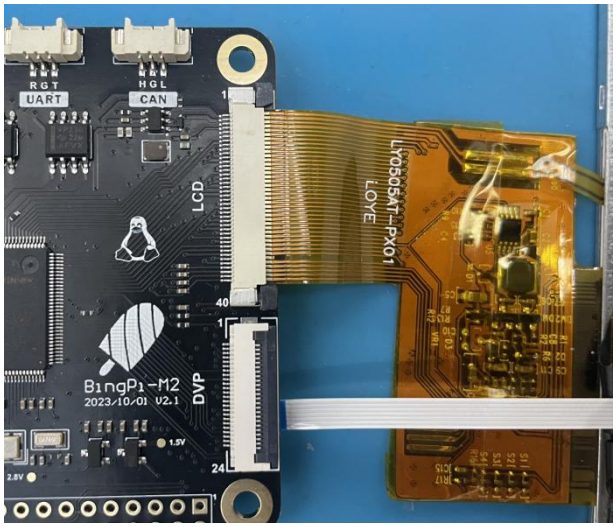
十四、RS-232 测试

RS-232 数据收发不需要额外的驱动，直接操作 ttyS4 即可，这里我们可以参考 RS-485 中应用程序编译，然后将可执行文件拷入开发板中运行测试，不再赘述。

十五、5 寸 800*480 LCD 显示屏适配

1、屏幕连接示意图





2、修改设备树文件

LCD 屏幕的驱动内核中已经处理好，所以如果想要适配 5 寸 800*480 分辨率的屏幕这里只需要修改一下 SDK 中的设备树文件即可，由于我们在 uboot 阶段显示了 logo 信息，因此除了内核中的设备树需要修改外，uboot 中的设备树也需要修改。

前面我们已经介绍了两个设备树在 SDK 中的位置，即 SDK 中

device/config/chips/t113/configs/bingpi_m2/board.dts & uboot-board.dts

board.dts 中已经为大家写好 800*480 分辨率的设备树内容，我们只需要屏蔽其他屏幕的内

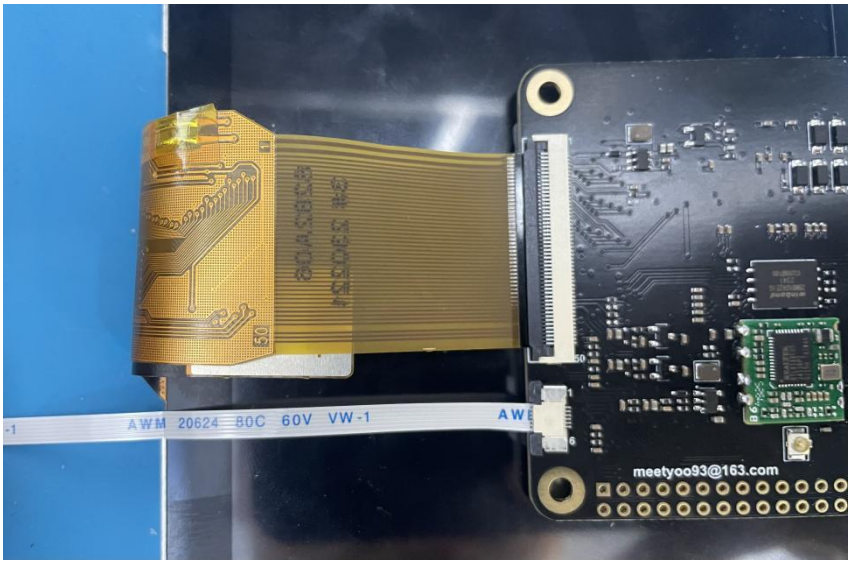
容，打开 800*480 配置的内容即可，同时将整段的 lcd0 配置内容复制到 uboot-board.dts 中替换原来的内容即可。

完成以上修改后，执行 make && mboot && pack 命令，即可生成适配 5 寸屏 800*480 分辨率的镜像。

更多屏幕的适配流程，可以资料包驱动开发笔记文件夹，lcd 文件夹中的 Tina_Linux LCD 驱动移植指南文档。

十六、7 寸 1024*600 LCD 显示屏适配

1、屏幕连接示意图



2、修改设备树文件

LCD 屏幕的驱动内核中已经处理好，所以如果想要适配 7 寸 1024*600 分辨率的屏幕这里只需要修改一下 SDK 中的设备树文件即可，由于我们在 uboot 阶段显示了 logo 信息，因此除了内核中的设备树需要修改外，uboot 中的设备树也需要修改。

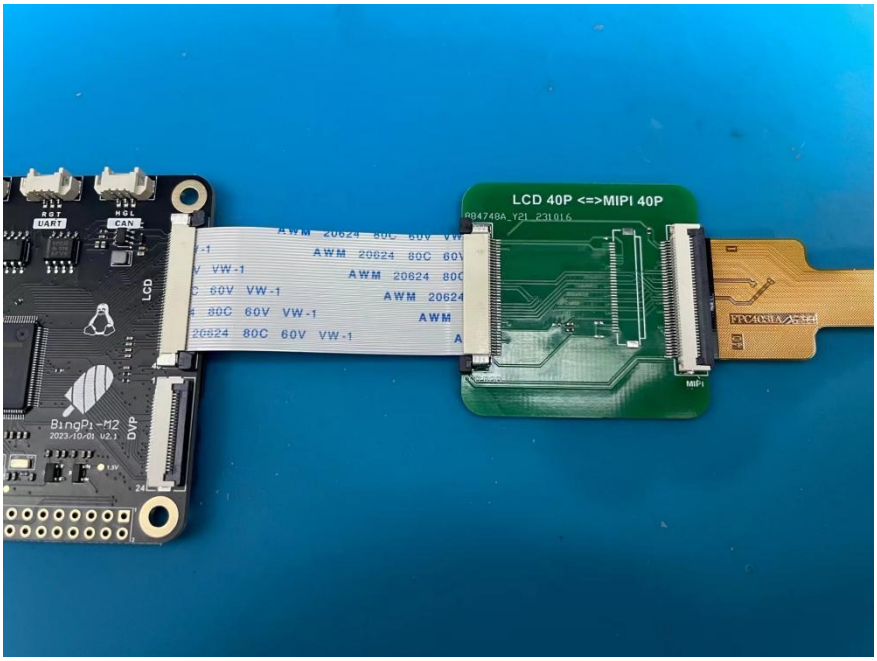
前面我们已经介绍了两个设备树在 SDK 中的位置，即 SDK 中 device/config/chips/t113/configs/bingpi_m2/board.dts & uboot-board.dts board.dts 中已经为大家写好 1024*600 分辨率的设备树内容，我们只需要屏蔽其他屏幕的内容，打开 1024*600 配置的内容即可，同时将整段的 lcd0 配置内容复制到 uboot-board.dts 中替换原来的内容即可。

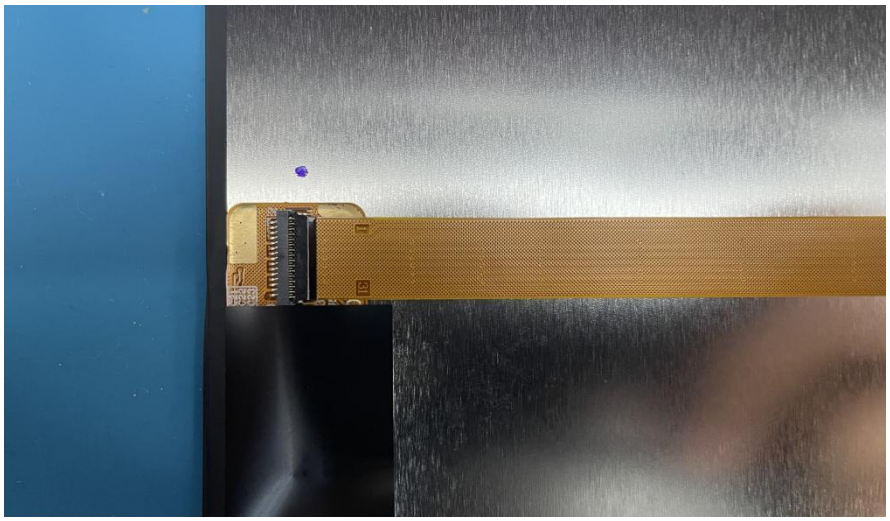
完成以上修改后，执行 make && mboot && pack 命令，即可生成适配 7 寸屏 1024*600 分辨率的镜像。

更多屏幕的适配流程，可以资料包驱动开发笔记文件夹，lcd 文件夹中的 Tina_Linux LCD 驱动移植指南文档。

十七、10.1 寸 800*1280 MIPI 显示屏适配

1、屏幕连接示意图





2、修改设备树文件

配套 MIPI 屏幕的驱动内核中已经处理好，所以如果想要适配 10.1 寸 800*1280 分辨率的屏幕这里只需要修改一下 SDK 中的设备树文件即可，由于我们在 uboot 阶段显示了 logo 信息，因此除了内核中的设备树需要修改外，uboot 中的设备树也需要修改。

前面我们已经介绍了两个设备树在 SDK 中的位置，即 SDK 中 `device/config/chips/t113/configs/bingpi_m2/board.dts & uboot-board.dts` `board.dts` 中已经为大家写好 `800*1280` 分辨率的设备树内容，我们只需要屏蔽其他屏幕的内容，打开 `800*1280` 配置的内容即可，同时将整段的 `lcd0` 配置内容复制到 `uboot-board.dts` 中替换原来的内容即可。

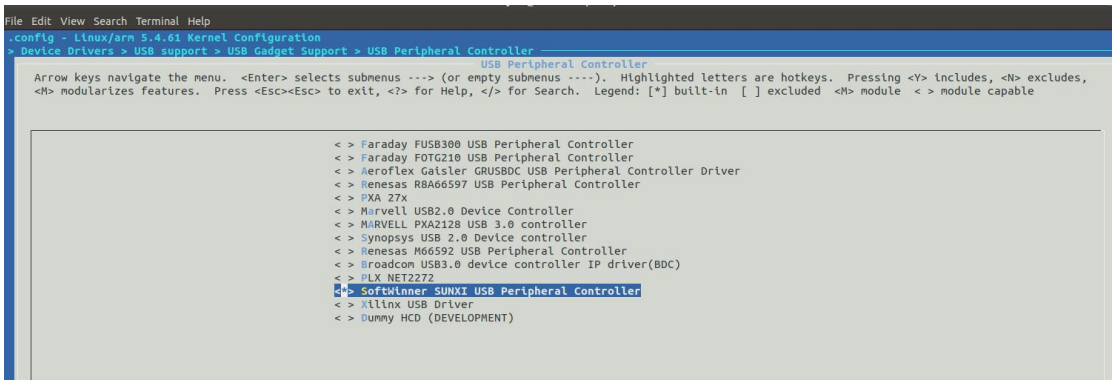
`10.1` 寸屏，设备树中的触摸部分也需要修改，搜索直接找到触摸 `gt9xx` 部分，根据屏幕后面标签，直接选择相应部分打开，然后屏幕其他部分。

完成以上修改后，执行 `make && mboot && pack` 命令，即可生成适配 `10.1` 寸屏 `800*1280` 分辨率的镜像。

更多屏幕的适配流程，可以资料包驱动开发笔记文件夹，lcd 文件夹中的 Tina_Linux LCD 驱动移植指南文档。

十八、开启 ADB

执行 make kernel_menuconfig 命令，打开内核如下配置。



十九、启动 logo 修改

系统启动 logo 为 bootlogo.bmp 文件，所在位置 target/allwinner/generic/boot-resource/boot-resource/bootlogo.bmp 只需要替换 bootlogo.bmp 文件，重新打包(pack)即可完成启动 logo 更换。