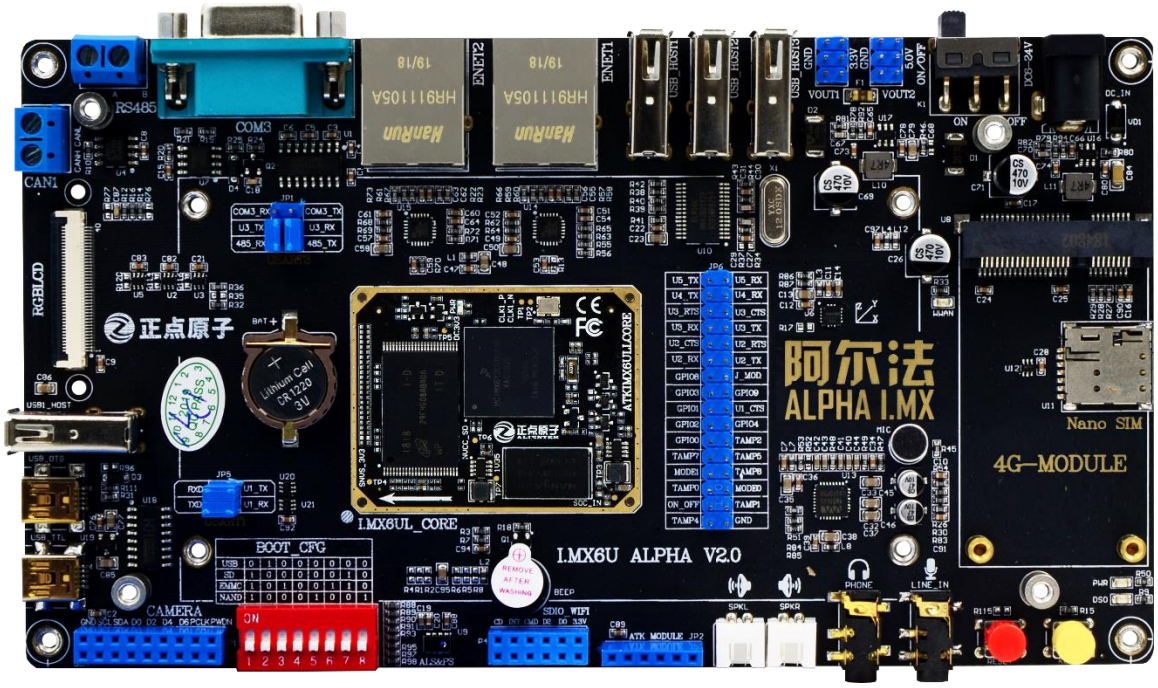
I.MX6U 用户快速体验 V1.9

-I.MX6U-ALPHA/MINI 快速体验



正点原子公司名称 ：广州市星翼电子科技有限公司

原子哥在线教学平台：www.yuanzige.com

开源电子网 / 论坛 ：http://www.openedv.com/forum.php

正点原子淘宝店铺 ：https://openedv.taobao.com

正点原子官方网站 ：www.alientek.com

正点原子 B 站视频 ：https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿：	正点原子 linux 团队	正点原子 linux 团队	2019.10.26
V1.1	1. 更新 2.2.1.1 小节，修改文档 vbs 脚本名称与实际要操作的 vbs 脚本名称不对应。 2. 更新 2.2 小节，修改拔码截图和 otg 连接截图。 3. 更新 2.3 小节，修改为免输 root 登录。 4. 其他小修改。	正点原子 linux 团队	正点原子 linux 团队	2019.11.2
V1.2	1. 删除 3.3.2 小节的 lcd 校准，改为 lcd 触摸测试。（备注：触摸驱动已经修正上报的坐标，不需要使用 lcd 校准。） 2. 增加 RGB 转 HDMI 模块测试 3. 增加 RCB 转 VGA 模块测试	正点原子 linux 团队	正点原子 linux 团队	2020.1.15
V1.3	1. 添加 MINI 板支持的实验说明	正点原子 linux 团队	正点原子 linux 团队	2020.3.18
V1.4	1. 修改第 4.3 小节，添加 rgb 转 hdmi 模块与 rgb 转 vga 模块相关设备树的编译 2. 添加第 2.1.1 小节，windows 10 安装 ch340 的方法	正点原子 linux 团队	正点原子 linux 团队	2020.4.28
V1.5	1. 修改一些描述用语，及某些注意事项 2. 添加 MINI 板封面图	正点原子 linux 团队	正点原子 linux 团队	2020.6.28

V1.6	1. 添加 3.22 小节 RTL8188CUS usb wifi，所支持的内核版本，若用户使用的是 RTL8188CUS usb wifi，请自行更新到 V1.6 版本以上的出厂系统固件及内核。避免测试不成功。 2. 添加 2.2.1.1 烧写 SD 卡遇到的错误说明。 3. 重新排版，优化阅读体验。 4. 新增 3.27 和 3.28 小节，重写第四、第五章。	正点原子 linux 团队	正点原子 linux 团队	2020.11.16
V1.7	1. 修改 1.2 小节，添加 Uboot，内核和文件系统版本的历史记录。 2. 修改 3.16 和 3.17 小节。新增正点原子摄像头 ov5640 模块 RGB565 采集和 ffmpeg 的简单使用。新增正点原子摄像头 ov7725 与 ov2640 模块的支持及使用。	正点原子 linux 团队	正点原子 linux 团队	2021.01.30
V1.7.1	1. 更新 1.2 小节的文件系统历史记录。 2. 更新 5.6 小节常用软件版本。	正点原子 linux 团队	正点原子 linux 团队	2021.02.23
V1.7.2	1. 增加 3.29 小节蓝牙测试。	正点原子 linux 团队	正点原子 linux 团队	2021.03.09
V1.7.3	1. 更新 1.2.2 小节的 uboot、内核和文件系统历史版本。 2. 修改 3.27 小节，由于内核开启了 apmd 电源管理，需要休眠则先杀死 apmd 程序。 3. 部分小修改含格式修改。	正点原子 linux 团队	正点原子 linux 团队	2021.4.14
V1.7.4	1. 添加 3.30 和 3.31 章节。 2. 其他小修改。	正点原子 linux 团队	正点原子 linux 团队	2021.6.18
V1.8	1. 优化 3.24 小节 sdio wifi 测试步骤，添加 sdio 开启热点和桥接的功能。 2. 优化 3.29 小节蓝牙测试步骤。 3. 优化 3.23 小节 4G 模块测试。 4. 添加 3.18 小节 EC20 ppp 拨号上网测试方法，同时优化 EC20 测试方法。	正点原子 linux 团队	正点原子 linux 团队	2021.8. 4

v1.9	1. 更新 3.23 小节 me3630，ecm 固件上网 2. 更新 5.6 小节常用软件版本	正点原子 linux 团队	正点原子 linux 团队	2021.11. 05

目录

前言 10
第一章 ..... ATK I.MX6U 软硬件资源简介 .11
1.1 硬件资源简介 12
1.1.1 I.MX6U-ALPHA 开发板底板资源简介 12
1.1.2 I.MX6U-Mini 开发板底板资源简介 13
1.1.3 I.MX6U 核心板资源 14
1.2 软件资源简介 14
1.2.1 软件资源一览表 14
1.2.2 出厂系统软件版本历史 15
第二章 ..... ATK I.MX6U 使用前准备 19
2.1 安装驱动和串口调试终端软件 20
2.1.1 安装 CH340 驱动 20
2.1.2 如何使用串口调试终端软件 20
2.2 固化系统 22
2.2.1 使用 mfgtool 上位机固化系统（OTG 方式） 22
2.2.2 使用脚本固化系统 27
2.3 登录开发板 33
第三章 ATK I.MX6U 功能测试 34
3.1. LED 与蜂鸣器测试 35
3.2 按键测试 35
3.3 LCD 触摸屏 36
3.3.1 LCD 背光调节 36
3.3.2 LCD 触摸测试 37
3.3.3 LCD 显示控制 38
3.4 串口测试 38
3.4.1 RS232 串口测试 38
3.4.2 RS485 串口测试 40
3.5 DDR 测试 40
3.6 TF(SD)卡读写测试 41
3.6.1 TF(SD)卡写速度测试 41
3.6.2 TF(SD)卡读速度测试 41

3.7 NAND FLASH 读写速度测试 42
3.8 系统时钟与 RTC 时钟 43
3.9 查看系统信息 44
3.10 温度传感器 45
3.11 网口测试 45
3.12 FlexCAN 测试 49
3.13 USB 接口测试 50
3.13.1 HOST 模式读写测试 50
3.13.2 DEVICE 模式测试 52
3.13.3 USB SERIAL 测试 52
3.14 USB 鼠标测试 53
3.15 音频测试 53
3.15.1 ALSA 简单使用 53
3.15.2 Headphone 测试 54
3.15.3 Speaker 测试 55
3.15.4 LINE IN 音频输入测试 55
3.15.5 MIC IN 录音测试 56
3.16 ov5640/ov2640/ov7725 摄像头测试 57
3.16.1 ov5640 摄像头 57
3.16.2 ov2640 摄像头 62
3.16.3 ov7725 摄像头 65
3.17 USB 摄像头测试 69
3.17.1 使用 gstreamer 采集 69
3.17.2 使用 ffmpeg 采集 70
3.18 EC20 4G 模块上网测试 70
3.18.1 ppp 拨号上网 73
3.18.2 使用 quectel-CM 75
3.19 视频播放测试 77
3.20 AP3216C 测试 78
3.21 icm20608 测试 79
3.22 USB WIFI 模块测试 80
3.22.1 Station（上网）模式 82
3.22.2 SoftAP（热点）模式 83

3.22.3 Bridge（桥接）模式 85
3.23 ME3630-W 4G 模块测试 87
3.23.1 pppd 拨号上网 88
3.23.2 通过 ECM 上网 90
3.24 SDIO WIFI 测试 92
3.24.1 Staion(上网) 模式 95
3.2.4.2 SoftAP(热点) 模式 95
3.24.3 Bridge(桥接)模式 95
3.25 RGB 转 HDMI 模块测试 95
3.26 RGB 转 VGA 模块测试 96
3.27 关机、重启和休眠 96
3.28 查看 CPU 主频 97
3.29 USB 蓝牙测试 98
3.29.1 蓝牙建立连接 100
3.29.2 蓝牙音乐 103
3.29.3 蓝牙传送文件 103
3.29.4 更多蓝牙工具 104
3.30 DS18B20 测试 105
3.31 DHT11 测试 107
第四章 ATK I.MX6U 交叉编译 111
4.1 安装通用 ARM 交叉编译工具链 112
4.2 安装 Poky 交叉编译工具链 112
4.3 编译出厂源码 U-boot 113
4.4 编译出厂源码内核及模块 114
4.6 编译出厂 Qt GUI 综合 Demo 115
4.7 编译一个简单的 c 文件 116
第五章 ATK I.MX6U 文件系统功能简介 118
5.1 文件系统目录简介 119
5.2 查看 Qt 环境变量 119
5.3 如何禁用 Qt 桌面启动 120
5.4 如何创建自启动程序 120
5.5 如何设置静态 ip 120
5.6 系统常用软件版本 121

附录 A 121

前言

写这个用户体验是为了帮助用户能够对板子开发板快速上手，请使用出厂系统去证以下功能，不能使用其他系统如自己移植的 busybox 文件系统（不保证完全适用！因为 busybox 这些是简易文件系统，没有正点原子制作的 Yocto 文件系统功能完整，正点原子出厂系统体积比较大，因为要添加的功能越来越多，如果需要做轻量级的系统，出厂系统并不适用）可参考以下文档验证板子的功能。是用户体验文档，也可以说是非常丰富的硬件测试文档，许多实验都需要保证硬件正常的情况下才继续进行，在这里，您将学习到 Linux 板子常用的硬件测试方法及某些 linux 指令的使用方法。

第一章 ATK I.MX6U 软硬件资源简介

本章介绍的内容有如下。

- 正点原子 I.MX6U 开发板硬件介绍。让用户了解开发板有哪些可用资源。
- 正点原子 I.MX6U 出厂系统版本介绍。介绍出厂系统软件版本，方便用户知道手头上的出厂系统软件版本状态，及时更新，可体验更多优质功能。

1.1 硬件资源简介

正点原子目前已经拥有多款 STM32、I.MXRT 以及 FPGA 开发板，这些开发板常年稳居淘宝销量冠军，累计出货超过 10W 套。这款 ALPHA 开发板，是正点原子推出的第一款 Linux 开发板，采用底板+核心板的形式。接下来我们分别介绍 I.MX6U-ALPHA 开发板的底板和核心板。

1.1.1 I.MX6U-ALPHA 开发板底板资源简介

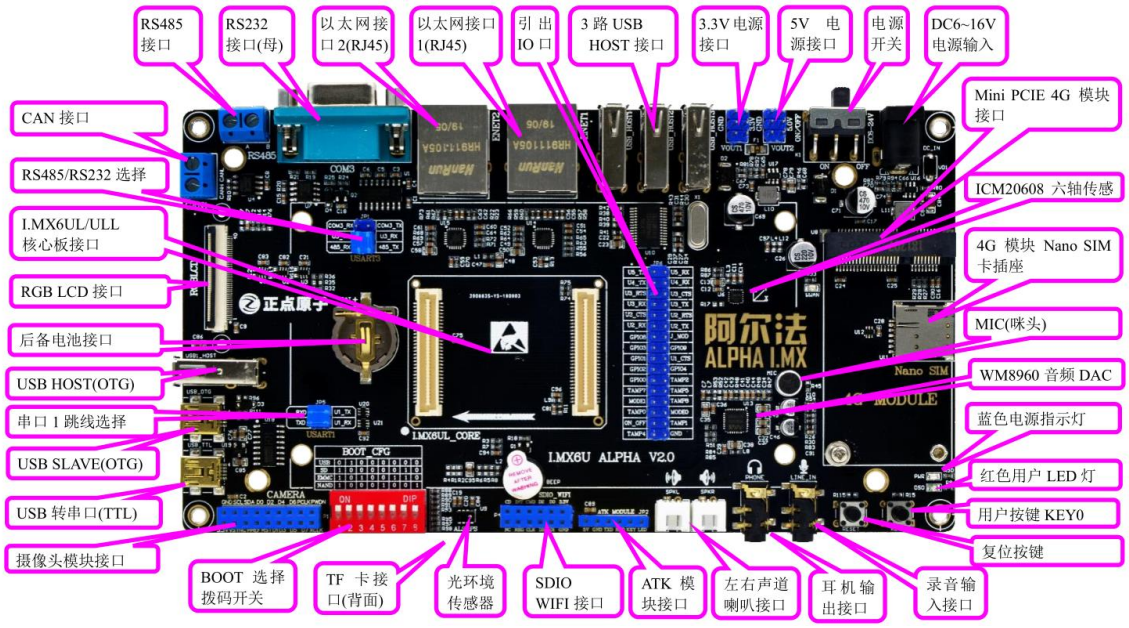


图 1.1.1 1 I.MX6U-ALPHA 开发板底板资源简介

详细请看我们的《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf》第五章 I.MX6U-ALPHA 开发板平台介绍的 5.1~5.2.1 小节。

◆1 个核心板接口，支持 I.MX6UL/6ULL 等核心板
◆ 1 个电源指示灯（蓝色）
◆ 1 个状态指示灯（红色）
◆ 1 个六轴（陀螺仪+加速度）传感器芯片，ICM20608
◆ 1 个高性能音频编解码芯片，WM8960
◆ 1 路 CAN 接口，采用 TJA1050 芯片
◆ 1 路 485 接口，采用 SP3485 芯片
◆ 1 路 RS232 串口（母）接口，采用 SP3232 芯片
◆ 1 个 ATK 模块接口，支持正点原子蓝牙/GPS/MPU6050/手势识别等模块
…

1.1.2 I.MX6U-Mini 开发板底板资源简介

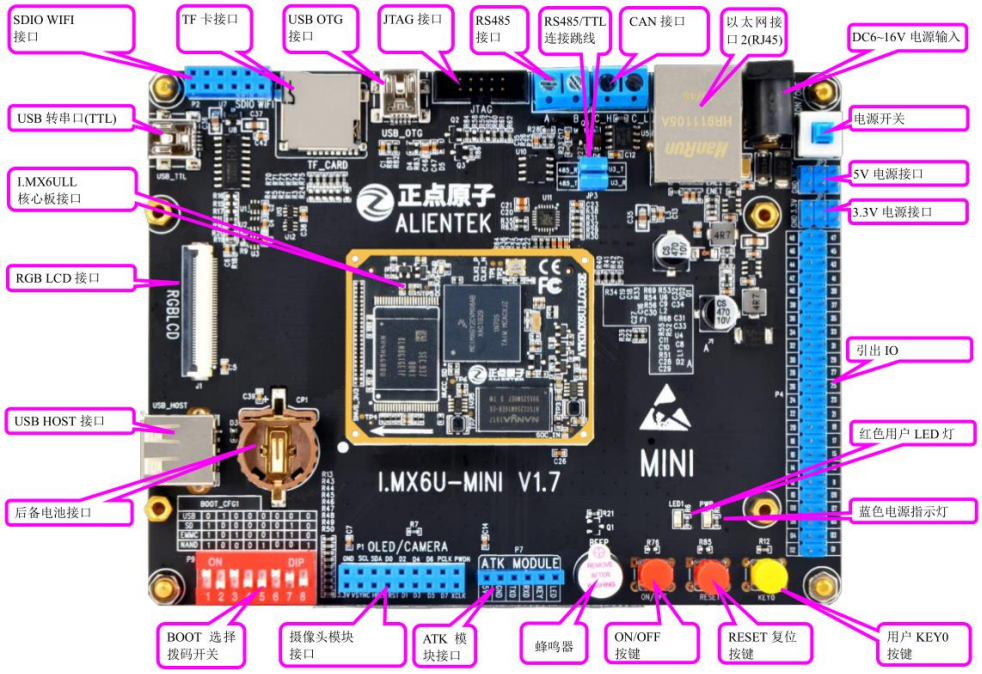


图 1.1.2 1 I.MX6U-Mini 开发板底板资源简介

- 1 个核心板接口，支持 I.MX6ULL 核心板。
- 1 个电源指示灯（蓝色）。
- 1 个状态指示灯（红色）。
- 1 路 CAN 接口，采用 TJA1050 芯片。
- 1 路 485 接口，采用 SP3485 芯片。
- 1 个 ATK 模块接口，支持正点原子蓝牙/GPS/MPU6050/手势识别等模块。
- 1 个摄像头模块接口。
- 1 个 USB 串口，可用于代码调试。
- 1 个 USB HOST 接口，用于 USB 主机通信。
- 1 个有源蜂鸣器 。
- 1 个 RS232/RS485 选择接口。
- 1 个 TF 卡接口。
- 1 个 10M/100M 以太网接口（RJ45）

…

详细请看我们的《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf》第五章 I.MX6U-ALPHA 开发板平台介绍的 5.1~5.2.1 小节。

1.1.3 I.MX6U 核心板资源

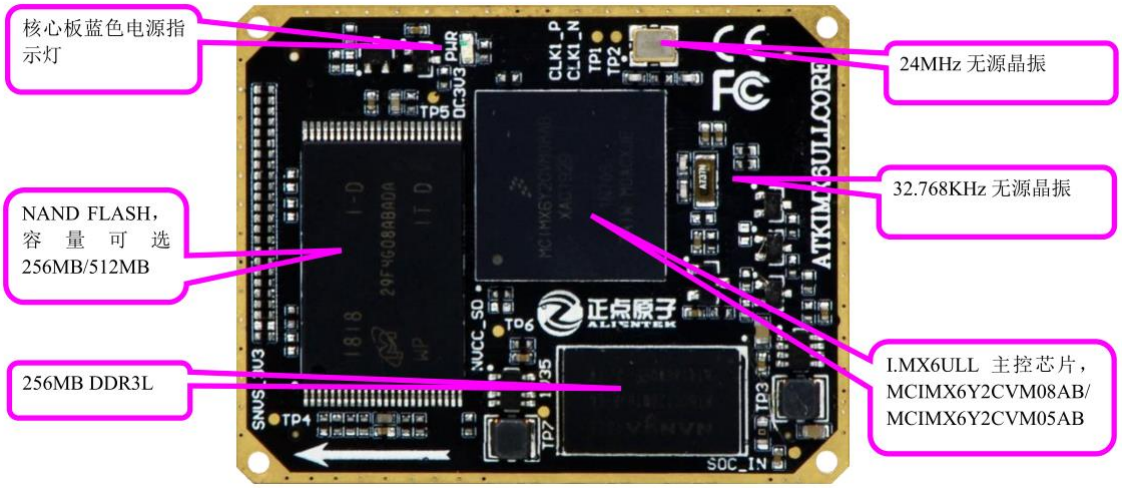


图 1.1.3 1 I.MX6ULL NAND BTB 接口核心板资源图

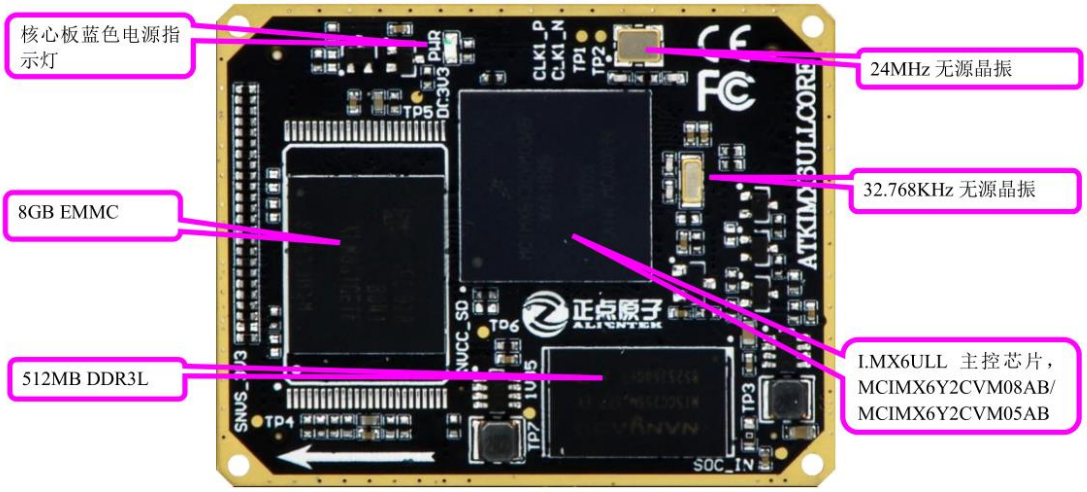


图 1.1.3 2 I.MX6ULL EMMC BTB 接口核心板资源图

详细请看我们的《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf》第五章 I.MX6U-ALPHA 开发板平台介绍的 5.1.3 小节。

1.2 软件资源简介

1.2.1 软件资源一览表

类型	描述	备注
uboot	Uboot 版本为 2016.03	提供源码。 支持 LCD 显示、支持 SD 卡和 EMMC、支持网络、支持 NANDFlash、支持环境变量修改等。
Linux 内核	内核版本为 4.1.15	提供源码
	提供 busybox、buildroot、yocto、	

根文件系统 rootfs	ubuntu 这四种根文件系统及其制作方法	提供详细的制作教程
交叉编译器	提供两种交叉编译器 （1）通用 ARM 交叉编译器 arm-linux-gnueaihf- 版本 4.9.4 （2）Poky 交叉编译器 arm-poky-linux-gnueabi- 版本 5.3.0	提供软件
Qt5 根文件系统	Qt 版本为 5.12.9（资料版本 V1.6 之前是 Qt 5.6.2）	提供详细的教程
系统烧写方法	MFGTOOL 和 SD 卡两种	提供详细的使用教程

…等等

详细请看我们的《【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf》第五章 I.MX6U-ALPHA 开发板平台介绍的 5.2.2 小节。

1.2.2 出厂系统软件版本历史

出厂 Uboot 版本在 Uboot 启动时会打印 Git 版本号，如下图。

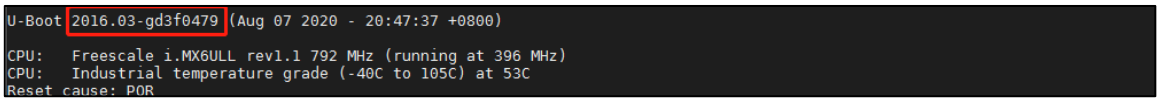


图 1.2.2 1 查看出厂系统里的 Uboot 版本

出厂 Uboot 历史版本			
版本	Git 版本	历史记录	日期
v1.0	g9bd38ef	无	2019.10.26
v1.1	gd9420c3	1.添加 ddr 校准参数	2019.11.28
v1.2	g4e04879	1.去掉 uboot 的 Logo 显示。 2.添加 rgb 转 vga 模块和 rgb 转 hdmi 模块识别设备树的代码支持。	2020.01.15
v1.3	g4475ea1	1.添加驱动指南教程的 uboot 配置文件，支持以驱动指南教程的方法编译出厂 uboot 源码。	2020.04.29
v1.4	gd3f0479	1.修复 uboot 读取 CPU 主频错误问题。 2.修改板卡名字为 I.MX6U ALPHA|MINI	2020.08.07
v1.5	ge468cdc	1.关闭 Uboot 中 LCD 的背光，避免某些屏开机瞬间白屏的现象。	2021.03.29
v1.6	gee88051	1.禁用 lcd。	2021.11.05

在进入内核时会打印内核的版本号，或者进入文件系统后，使用 `uname -r` 指令查看内核 Git 版本号，及编译的时间。

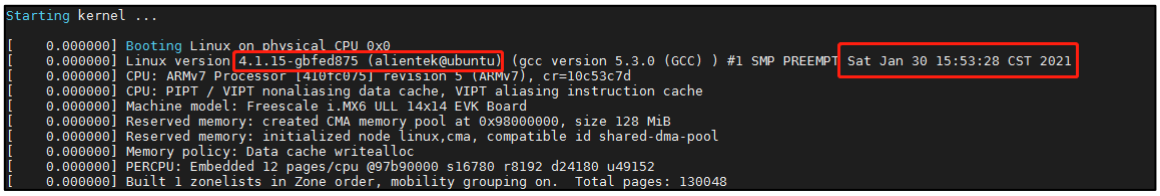


图 1.2.2 2 查看出厂内核版本

出厂内核历史版本			
版本	Git 版本	历史记录	日期
v1.0	g49efdaa	无	2019.10.26
v1.1	gbedf008	1.优化网络	2019.11.28
v1.2	g52f6b26	1.添加 rgb 转 vga 模块和 rgb 转 hdmi 模块识别设备树的代码支持。 2.修复 gt9xx 驱动和 edt-ft5x06 驱动上报坐标错误问题。	2020.01.05
v1.3	gb8ddbbc	1.添加 rtl_bt 蓝牙驱动支持。2.添加 rtl 8188cus 的支持 3.修复 Lan 870 phy 芯片输入时钟峰峰值超过额定值。	2020.04.29
v1.4	g8a704c6	1.修复 rtl 8182CU 驱动不能正常使用的问题 2.添加 RGB LCD 屏幕触控芯片 gt1151 的驱动支持。	2020.08.07
v1.5	g871ccc8	1.修复 4.3 寸屏（800x480, 480x272）的 RGB 屏 gt9147 系列触控芯片不能触摸的问题。维护 gt1151 的触摸驱动。	2020.12.23
v1.6	gbfed875	1.添加正点原子 ov5640 模块支持多种格式采集的功能及正点原子 ov2640、ov7725（不带 FIFO）摄像头模块的支持。	2021.01.30
v1.7	gad512fa	1.优化 Uboot 到内核阶段某些 LCD 闪屏的问题。	2021.03.29
v1.8	g14132a2	1.去除 ONOFF 功能。 2.设备树添加 DHT11 与 DS18B20 的配置。	2021.06.18
v1.9	gc0de9f6	1.开启 ch341 usb 转 serial 驱动。 2.开启 cp210x uart 驱动。	2021.07.22
v2.0	gc353ffb	1.优化 ov5640 驱动，匹配应用教程。 2.其他摄像头摄像优化。	2021.11.05

文件系统历史版本，在进入文件系统后，输入 cat /etc/version 即可查看历史版本。如下图。

图 1.2.2 3 查看出厂文件系统版本

出厂文件系统历史版本		
版本	历史记录	日期
v1.0	无	2019.10.26
v1.1	1.重新排版 qt 桌面，优化桌面滑动，写成精简桌面。取消各个程序下拉退出的功能，添加退出按钮。 2.修复可能不能录音的问题。 3.在文件系统/home/root/目录下添加测试所用的文件。 4.文件系统添加免输入 root 登录功能。	2019.11.28
v1.2	1.重新设计 Qt 桌面 2.修复 Qt 打开文件夹拖动文件导致系统不能启动的问题 3.添加 Qt 桌面子进程进场、出场动画 4.其他优化	2019.11.30
v1.3	1.修改开机界面 2.修复开机进入桌面程序时闪屏的问题 3.qt 界面的优化修改，详细请看 qt 综合例程源码处的说明 4.去除触摸坐标参考文件，修改 qt 运行时的环境变量	2020.01.15
v1.4	1.修改 Qt 桌面为第一版本时 Qt 桌面，添加音量控制	
v1.5	无	2020.11.14
v1.6	1.升级文件系统里的 Qt 版本为 Qt5.12.9，Qt 桌面重新打造，优化用户体验。文件系统的功能及资源请看 I.MX6U 快速体验第五章。	2020.11.18
v1.7	1.修复开机耳机默认没有音量的问题 2.修复 Qt 设置 App 里音量条 70%以下没有声音的问题 3.优化 Qt 相册滑动切换流畅度	2021.01.08
v1.8	1. 修复文件系统使用 Qt 无法显示 gif 图片的问题	2021.01.12
v1.9	1.添加 rsync 指令 2.添加 qt mysql 支持 3.添加正点原子 camera_settings ov5640_camera、ov772x_camera、ov2640_camera 应用程序 4.添加 mysql，sqlite3，cups，hplip 的支持	2021.01.30
v2.0	1.移除 mysql 2.其他优化	2021.2.23
v2.1	1.删除 UI 界面上的关机功能	2021.03.25

	2.添加若 USB WIFI 连接不上网络的提示。	
v2.2	1.添加 devmem2,memtool 工具 2.添加 Nginx web 服务器 3./home/root/driver/添加 ds18b20,dht11 驱动文件	2021.06.18
v2.3	1.添加 qt 支持 mysql 的 driver(默认不安装 Mysql,因为体积太大,需要额外安装包) 2.添加 pip,git 指令 3.添加 SDIO WIFI 测试脚本 /home/root/shell/wifi/alientek_sdio_wifi_setup.sh 4.添加 USB 蓝牙测试脚本 /home/root/shell/bluetooth/bluetooth_start.sh 5.优化网络，固定 MAC 地址，防止多次开机获取不同的 IP 6.添加 USB bluetooth 测试脚本 7.优化 SDIO WIFI 与 USB WIFI 开启热点 8.优化串口终端显示颜色，保存指令记录 9.添加 SDIO WIFI 初始化脚本 /home/root/shell/wifi/alientek_sdio_wifi_init.sh	2021.07.22
v2.4	1. 添加 me3630 4G 模块 ecm 上网程序，此方式更方便上网。 2. 禁用看门狗脚本，防止 poweroff 后重启。	2021.11.05

第二章 ATK I.MX6U 使用前准备

本章简介如下。

- 安装开发板的串口调试驱动，及如何使用串口调试上位机。
- 如何固化出厂时的系统。分为脚本固化与上位机固化。
- 登录开发板。

2.1 安装驱动和串口调试终端软件

2.1.1 安装 CH340 驱动

开发板的串口与电脑通信，需要安装CH340驱动，开发板光盘A-基础资料->3、软件->CH340驱动(USB串口驱动)_XP_WIN7共用找到SETUP.EXE双击进行安装。出现预安装成功或者安装成功则成功。如果Win10安装失败，请下载驱动精灵，先使用发货时提供的USB T字口数据线连接开发板底板的USB_TTL，USB接口端连接电脑，驱动精灵扫描硬件驱动，把相应的串口驱动装上就可以了。

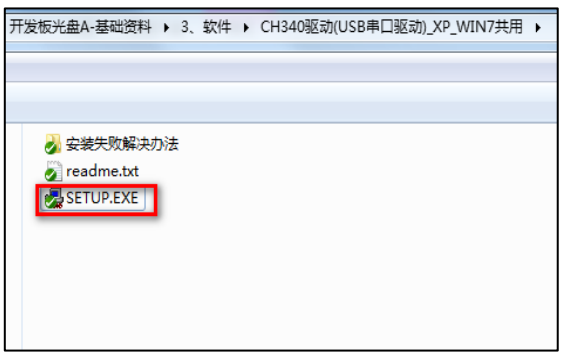


图 2.1.1 双击 SETUP.EXE 进行安装

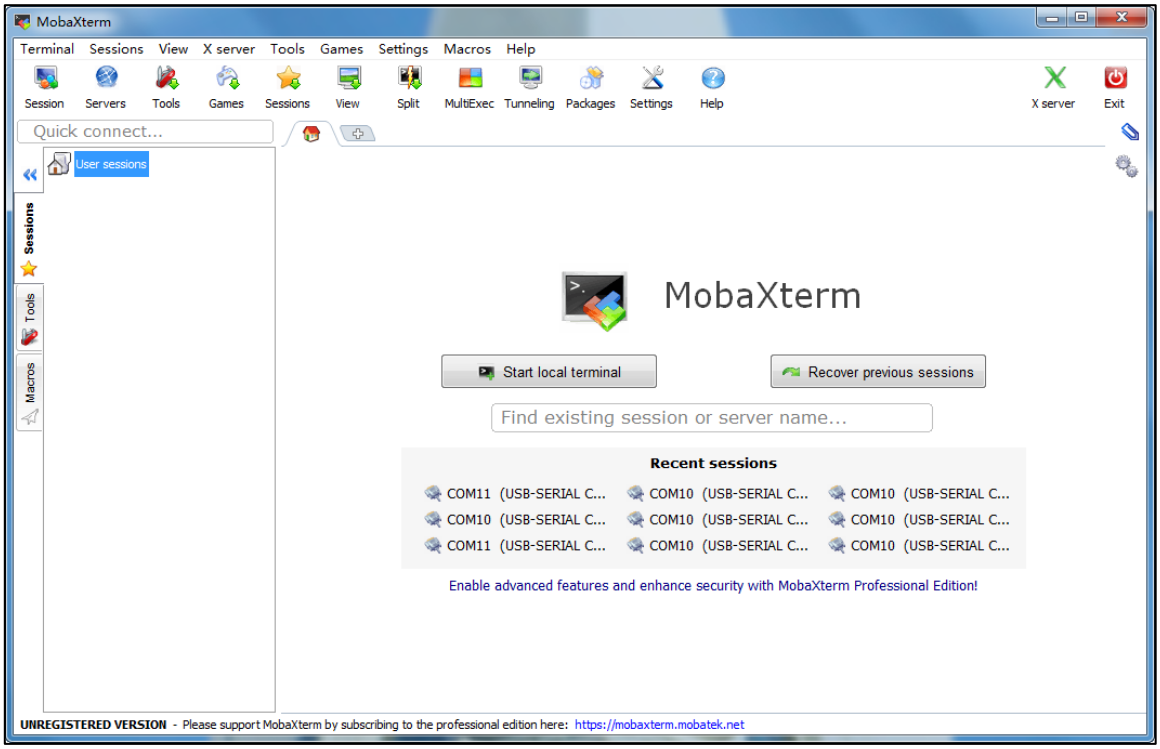
图 2.1.2 点击开始安装 CH340 驱动

2.1.2 如何使用串口调试终端软件

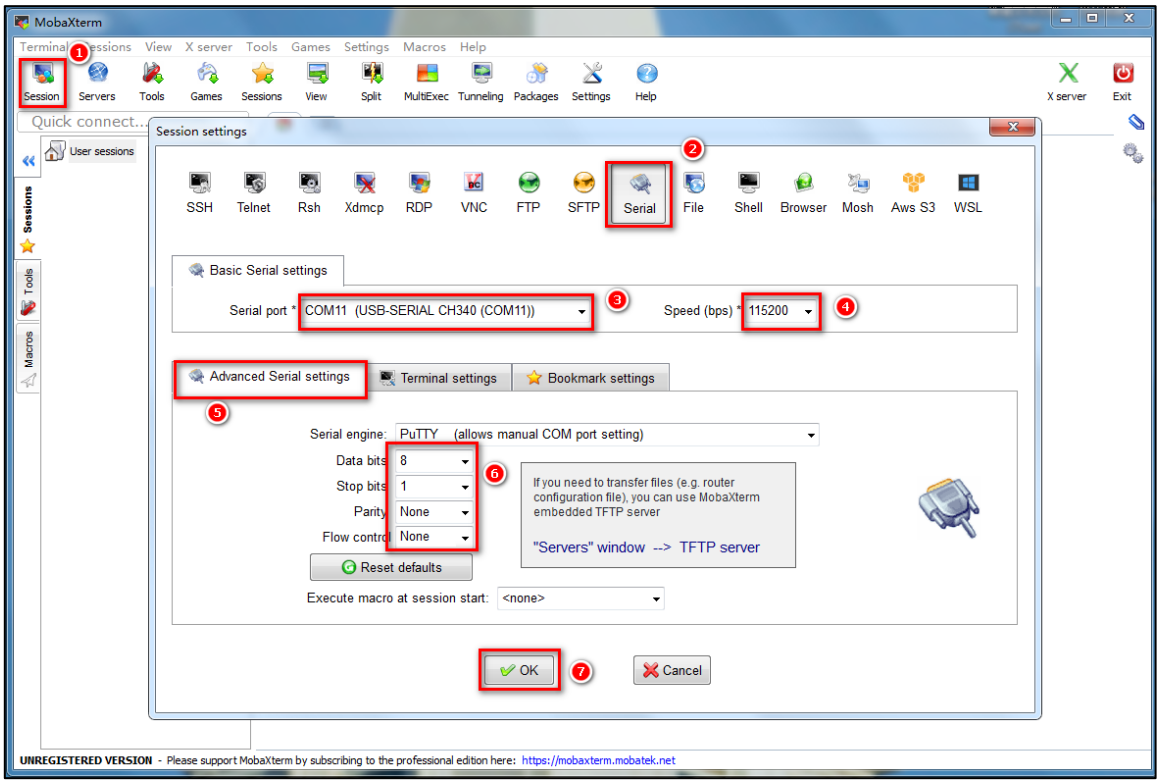
可以使用串口的终端软件有很多比如Xshell，SecureCRT，MobaXterm，甚至Ubuntu的终端都是可以连接开发板上的串口的。本文以MobaXterm安装为例，在众多终端软件来说算是对新手比较友好，因为界面比较好看，但是也有不足，不能自动重连串口。

在开发板光盘A-基础资料->3、软件-> MobaXterm_Installer_v12.3.zip双击解压进行安装即可，安装过程非常简单，默认下一步进行安装。这里不写安装过程了。

安装在桌面打开MobaXterm图标，软件界面如下图。

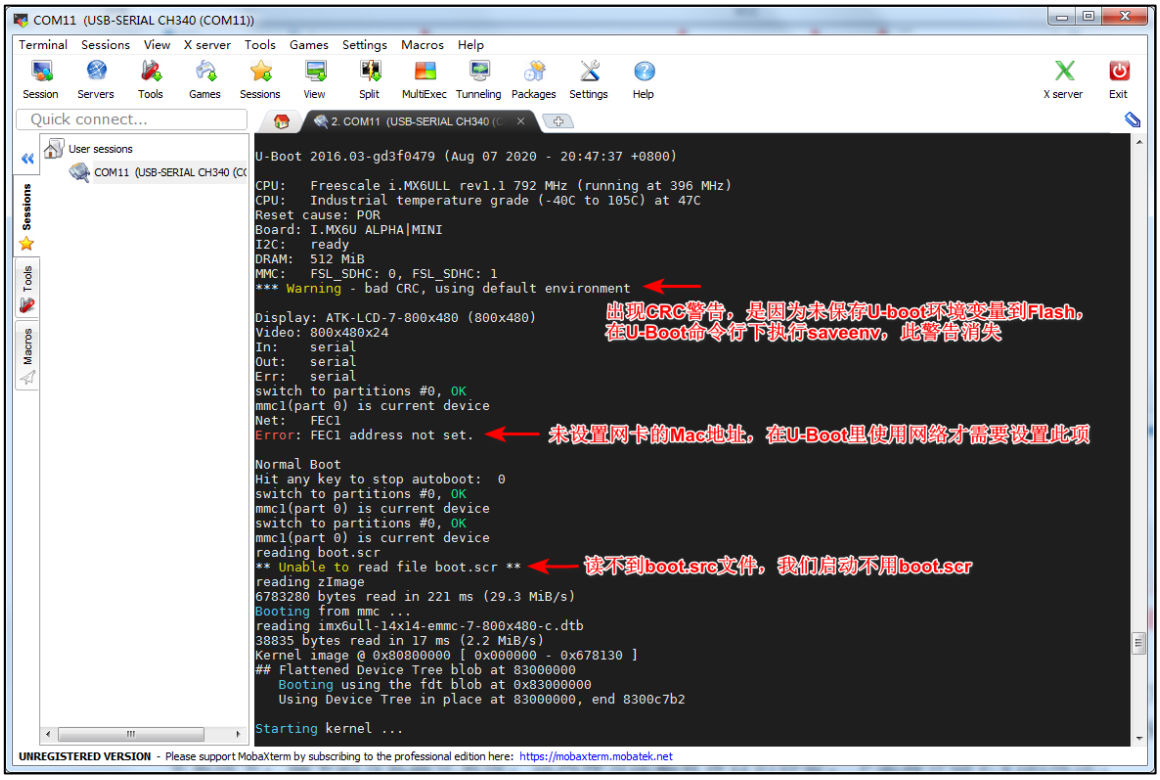


开发板接上 USB T 字口数据线到 USB_TTL，连接电脑，开发板插上电源，出货时拨码开关已经拨好。MobaXterm 按如下配置，点击 Session（会话）->选择 Serial（串口）类型->选择开发板设备对应的 com 口->选择波特率为 115200（开发板默认波特率为 115200）->高级设置->设置流控为 None->点击确认。



开发板上电，串口终端打印的信息如下。其中 U-Boot 阶段打印的信息如下，报的警告或者错误解释如图，新手常常担心打印信息出现错误，以为是什么毛病。这里我们作了解释。不过

Linux 里经常是需要打印信息，用来提示硬件初始化失败或用户操作不当等。



2.2 固化系统

开发板出厂已经固化系统到核心板上的 eMMC 或 NandFlash 存储介质里。建议自行更新系统固件（检查是不是最新系统，请看 1.2.2 小节），新的系统修复可能存在的 bug、优化程序及添加新的功能。如果不需要重新固化系统，请直接到 2.3 节登录开发板。

正点原子提供两种固化系统方式，第一种方式(NXP 官方提供的方法)是使用正点原子修改过的 NXP 官方的上位机工具 mfgtool。这种固化系统方式可以使用 PC 机在线直接固化系统。第二种方法（正点原子额外提供的方法）需要制作 TF 卡系统卡，插卡的方式固化系统。这两种方法都有各自的好处。下面将介绍它们的用法。

2.2.1 使用 mfgtool 上位机固化系统（OTG 方式）

以 ALPHA 底板为例，底板拨码开关 BOOT_CFG 设置如下，设置为 USB 连接方式，“1”代码 ON，“0”代表“OFF”。将拨码数字 2 处拨到 ON，其他的为 OFF。如下图

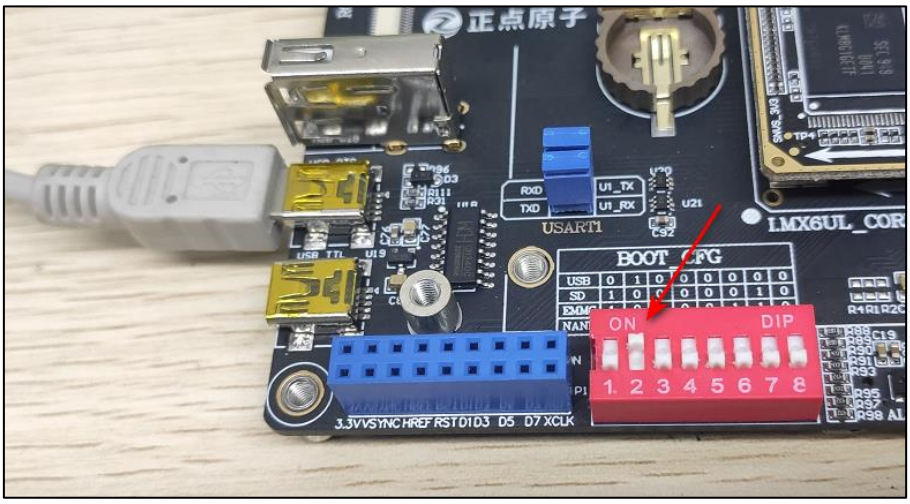


图 2.2.1 1 mfg 固化系统时底板的拨码开关设置

使用 USB 连接线连接底板的 USB_OTG 接口与 PC 机（电脑）。(注：MINI 底板的位置不一样。)

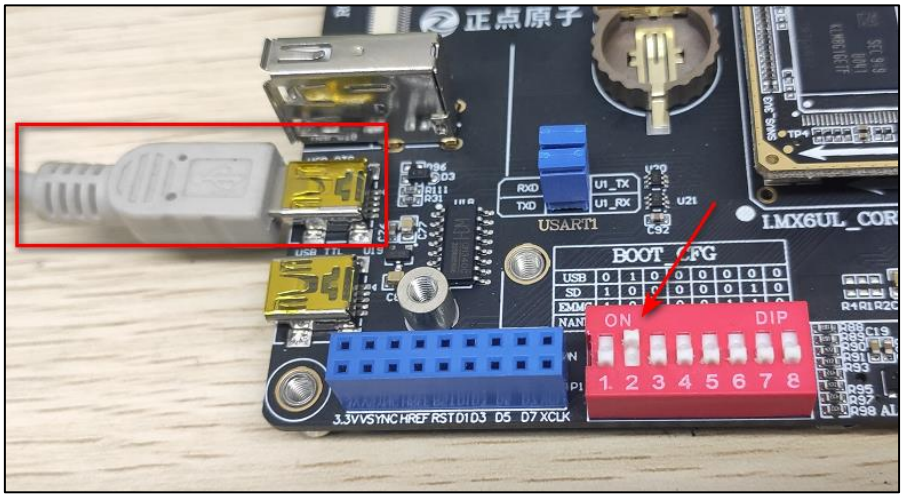


图 2.2.1 2 用串口线连 USB_OTG 接口

进入开发板光盘 A-基础资料->5、开发工具->4、正点原子 MFG_TOOL 出厂固件烧录工具->mfgtool 文件夹下，查看 vbs 脚本信息，如下图图 2.2.1.1 3。

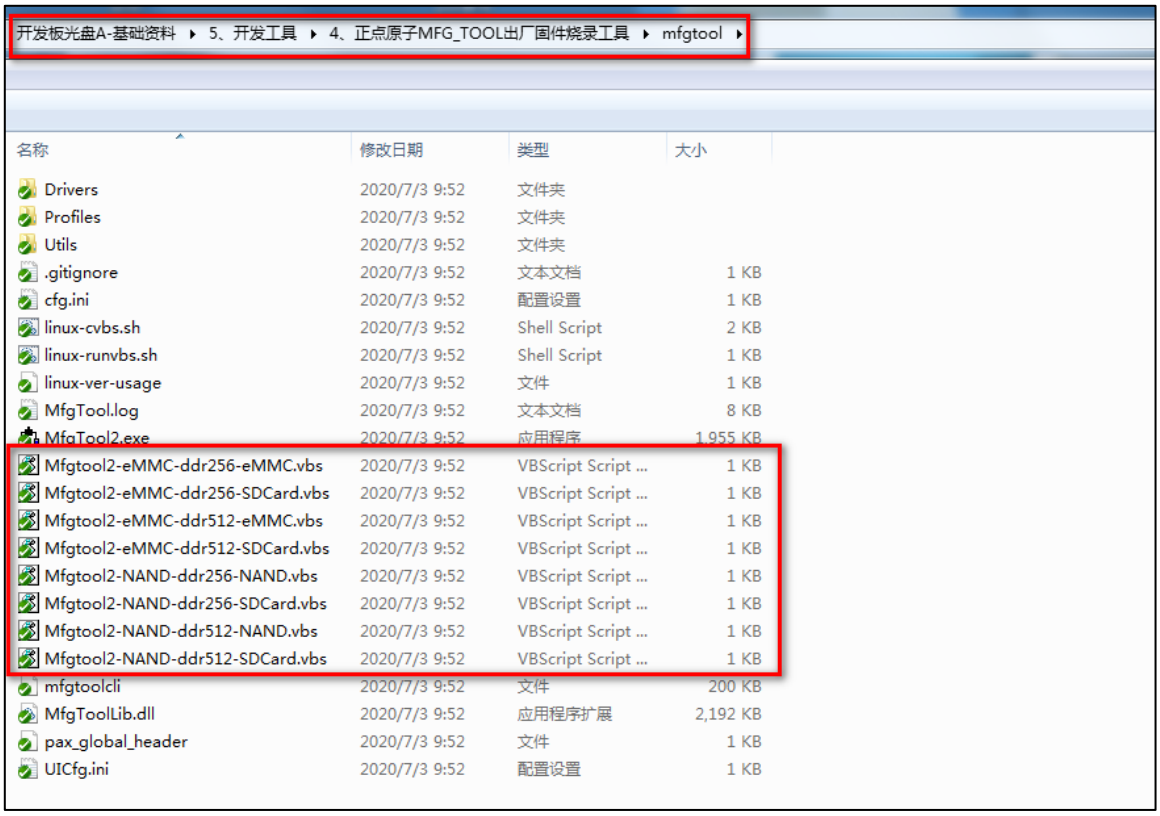


图 2.2.1 3 vbs 脚本信息

用户不需要知道这么多 vbs 脚本怎么用，其实我们也是只用到几个而已。目前正点原子在售的核心板类型只有两种，一种是 eMMC（8GB）核心板类型，DDR 大小为 512MB。另一种是 NandFlash（512MB），DDR 大小为 256MB。vbs 脚本解释如下:

以 Mfgtool2-eMMC-ddr512-eMMC.vbs 为例。

-eMMC：代表核心板上的存储介质为 eMMC

-ddr512：指核心板上的 DDR 大小为 512MB

-eMMC：选择此脚本是把固件烧写到 eMMC 上

2.2.1.1 固化系统到 TF 卡

用户需要准备一张 TF 卡，按前面的 vbs 脚本命名解释，比如要固化系统到 TF(SDCard)卡，那么需要确认用户手上的核心板类型是 eMMC 还是 NandFlash。比如现在用户手上核心板类型为 eMMC 类型核心板。所以我们选择 Mfgtool2-eMMC-ddr512-SDCard.vbs 这个脚本来烧写 ，如果是用户手上核心板类型 NandFlash 核心板，则选择 Mfgtool2-NAND-ddr256-SDCard.vbs。

比如编者手上的是 eMMC 核心板，要固化系统到 TF 卡上，按 2.2.1 小节拨码及连接方式，开发板上电，双击 Mfgtool2-eMMC-ddr512-SDCard.vbs。如果是第一次使用开发板 OTG 连接 PC 机（电脑），需要等待 PC（电脑）机自动安装驱动。点击右下角正在安装驱动的小图标，不要让 Winodws 去从 Windows update 里去找驱动，点击跳过，驱动自然会装上。否则用了 Windows update 里的驱动，可能在烧写过程中会卡住，不往下跑。等待安装驱动完成后，mfgtool 上位机界面会提示已经连接到设备 HID-compliant device，注意请不要打开虚拟机，如果虚拟机正在开启，OTG 会连接到虚拟机上去。在烧写时，如果遇到杀毒报错，建议关闭杀毒软件！

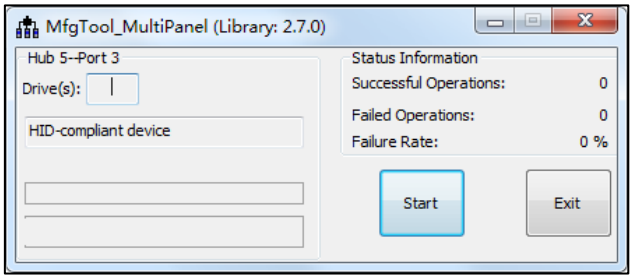


图 2.2.1.1 2 mfg 上位机连接开发板设备界面

此时将 TF 卡插入卡槽（需开发板先上电后再将 TF 卡插进卡槽，否则上电时会检测 TF 卡，这样 mfgtool 会连接不到开发板设备），如下图插上 SD 卡（MINI 底板卡槽的位置不一样）。

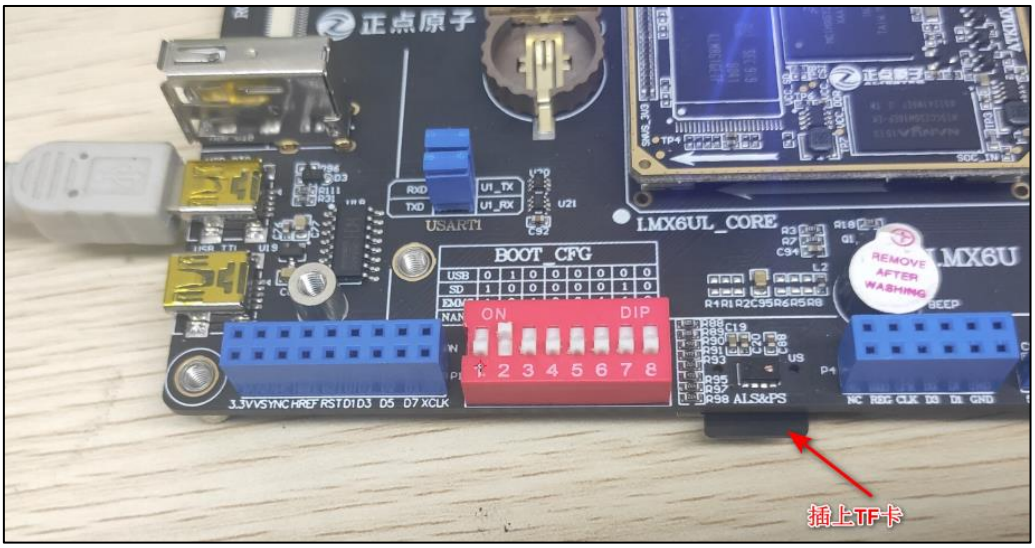


图 2.2.1.1 3 SD 卡插槽截图

直接点击 mfgtool 的 Start 按钮进行固化系统到 SD 卡。下图为点击 Start 按钮后的截图。固化系统到 SD 卡需要几分钟时间，请耐心等待。

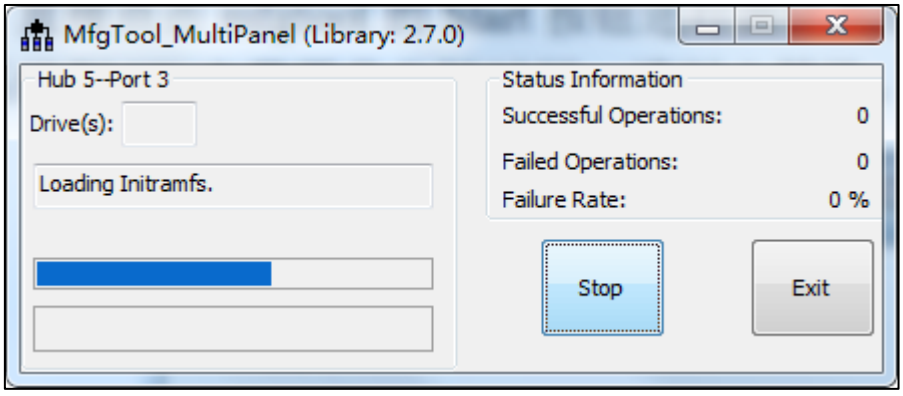


图 2.2.1.1 4 点击 Start 按钮后截图

下图图 2.2.1.1 5 为固化过程中的一步，正在写入文件系统

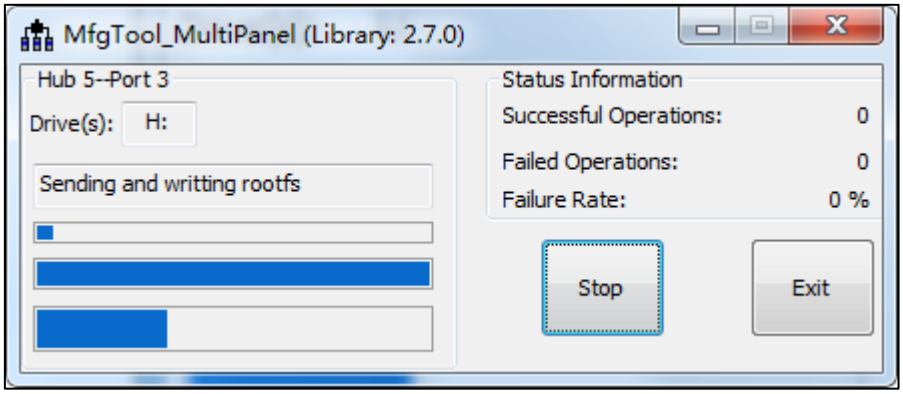


图 2.2.1.1 5 正在写入文件系统

固化完成如下图，点击 Stop 后再点 Exit 退出 mfgtool 上位机软件即可。

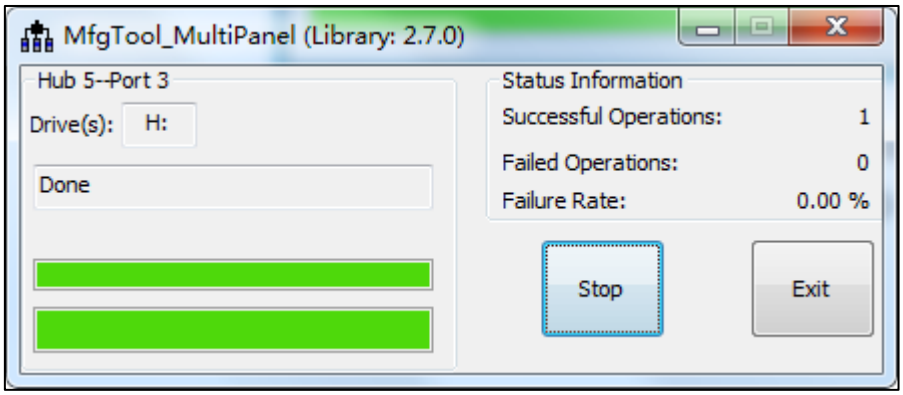


图 2.2.1.1 6 固化完成

测试从 SD 卡启动系统，拨码开关拨至 SD 卡启动方式 10000010，启动系统即可。

2.2.1.2 固化系统到 eMMC

使用前提:用户核心板类型带 eMMC 存储介质。目前原子在售的 eMMC 核心板 ddr 大小是 512MB。请双击 Mfgtool2-eMMC-ddr512-eMMC.vbs 这个 vbs 脚本文件进行固化。

请参考 2.2.1 1 小节固化系统到 SD 卡的步骤（注：固化时不要插入 SD 卡），选择固化到 eMMC 的 vbs 文件，固化完成后，将拨码开关拨至 eMMC 启动方式 10100110，启动系统即可。

2.2.1.3 固化系统到 NAND FLASH

使用前提:用户核心板类型带 NAND FLASH 存储介质。目前原子在售的 NAND FLASH 核心板都是 ddr 大小是 256MB。请双击 Mfgtool2-NAND-ddr256-NAND.vbs 这个 vbs 脚本文件进行固化。

请参考 2.2.1.1 小节固化系统到 SD 卡的步骤（注：固化时不要插入 SD 卡），选择固化到 NAND FLASH 的 vbs 文件，固化完成后，将拨码开关拨至 NAND 启动方式 10001001，启动系统即可。

2.2.2 使用脚本固化系统

脚本固化系统一般可用于批量固化与升级系统，不像 mfgtool 上位机那样还需要 PC 机和 USB T 字口数据线，且每次只能打开一个 mfgtool 上位机，用户可以自行修改好固化系统脚本，进行自动化固化测试，那么可以无需专业人员参与，即可批量固化系统。

脚本由正点原子编写提供。

2.2.2.1 固化系统到 TF(SD)卡

拷贝开发板光盘 A-基础资料->5、开发工具->4、正点原子 MFG_TOOL 出厂固件烧录工具->mfgtool->Profiles->Linux->OS Firmware->files 整个文件夹到 Ubuntu 虚拟机，如下图图 2.2.2.1 1，本文档已经拷贝 files 文件夹到 Ubuntu 虚拟机。

```
alientek@ubuntu:~/files$ ls
boot  filesystem  imx6mkemmcboot.sh  imx6mknandboot.sh  imx6mksdboot.sh  modules  README.txt  必读说明.txt
alientek@ubuntu:~/files$
```

图 2.2.2.1 1 拷贝 files 文件夹到 Ubuntu

使用 chmod 指令修改固化 TF 卡系统脚本 imx6mksdboot.sh 的权限

```
chmod +x imx6mksdboot.sh
```

```
alientek@ubuntu:~/files$ chmod +x imx6mksdboot.sh
alientek@ubuntu:~/files$ ls
boot  filesystem  imx6mkemmcboot.sh  imx6mknandboot.sh  imx6mksdboot.sh  modules  README.txt  必读说明.txt
alientek@ubuntu:~/files$
```

图 2.2.2.1 2 赋予脚本可执行权限

TF 卡用读卡器插到 Ubuntu 虚拟机，如果 Ubuntu 没提示连接可移动设备连接到虚拟机，按以下步骤连接到虚拟机。

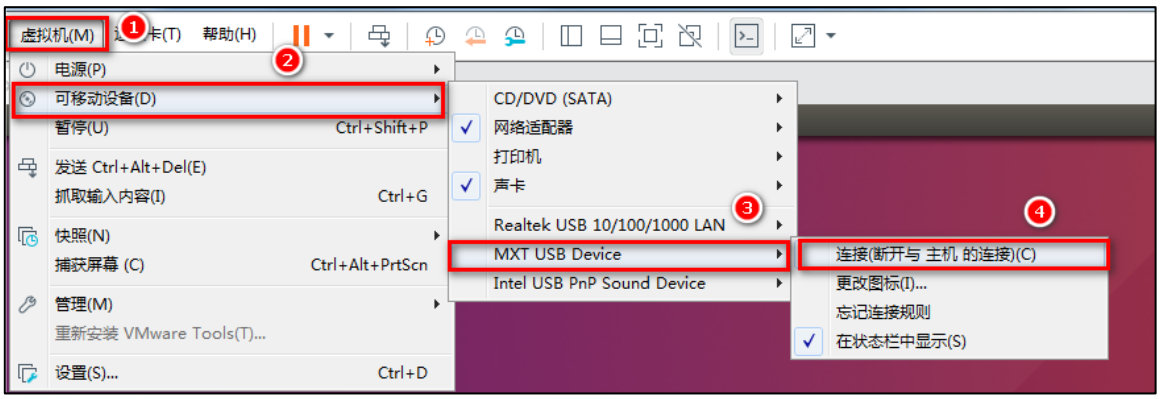


图 2.2.2.1 3 连接 TF 卡的步骤

输入如下指令查看 SD 卡挂载节点，如下图，编者的 SD 卡容量是 14.9GB(16GB)，可以看到挂载的节点为/dev/sdb。

```
sudo fdisk -l
```

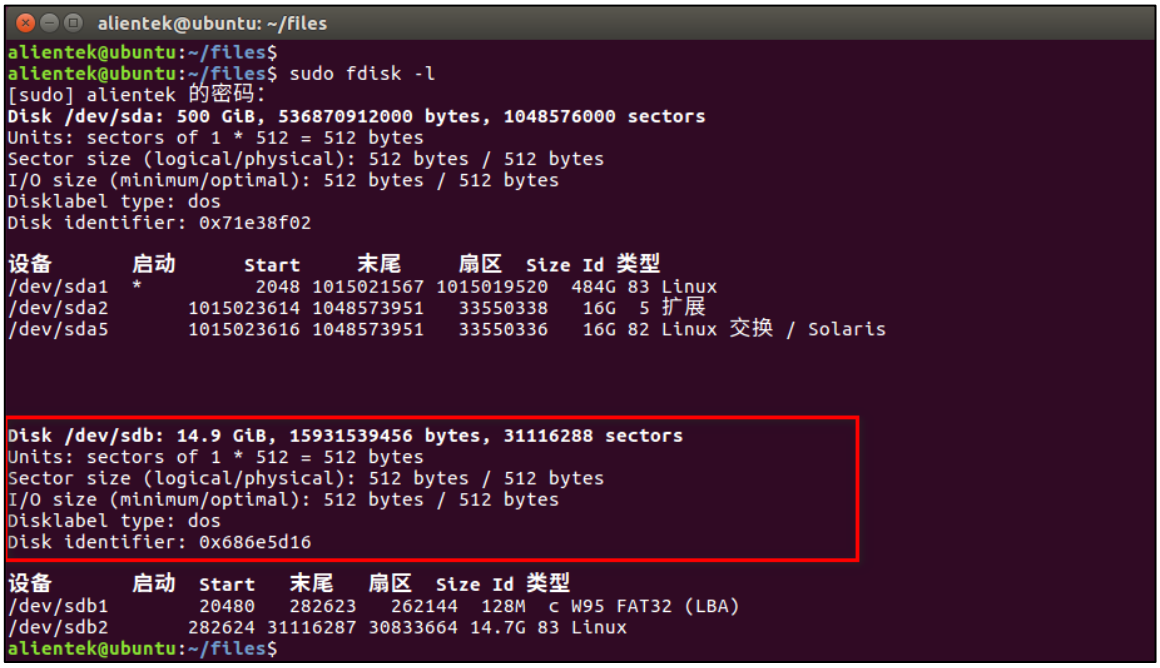


图 2.2.2.1 4 查看 U 盘在 Ubuntu 上的连接节点

执行./imx6mksdboot.sh --help 查看脚本的使用方法。

```
./imx6mksdboot.sh --help
```

图 2.2.2.1 5 查看 imx6mksdboot.sh 使用说明

用法说明：

用法: imx6mksdboot.sh [选项] <(必选)-device> <(可选)-flash> <(可选)-ddrsize>

（1） -device：指明设备节点（TF 卡挂载的节点如/dev/sdx），必需加这个参数

（2） -flash：指明核心板上的媒体存储介质，可选为（emmc|nand）

（3） -ddrsize：指明核心板上的 ddr 容量大小，可选为（512|256）MB

比如现在用户是核心板的 ddr 容量大小是 512MB，媒体存储介质是 eMMC。SD 卡挂载节点为/dev/sdb。那么固化 SD 卡的指令如下，执行指令后脚本会有中文再次询问 SD 卡所挂载的节点是否对应，将会清空 SD 卡上的所有数据，请注意备份重要的数据。按 Enter 键确认后继续，固化 SD 卡需要大约需要几分钟时间，这里根据个人电脑不同和所用 SD 卡不同，可能花费的时间差异比较大。

```
sudo ./imx6mksdboot.sh -device /dev/sdb -flash emmc -ddrsize 512
```

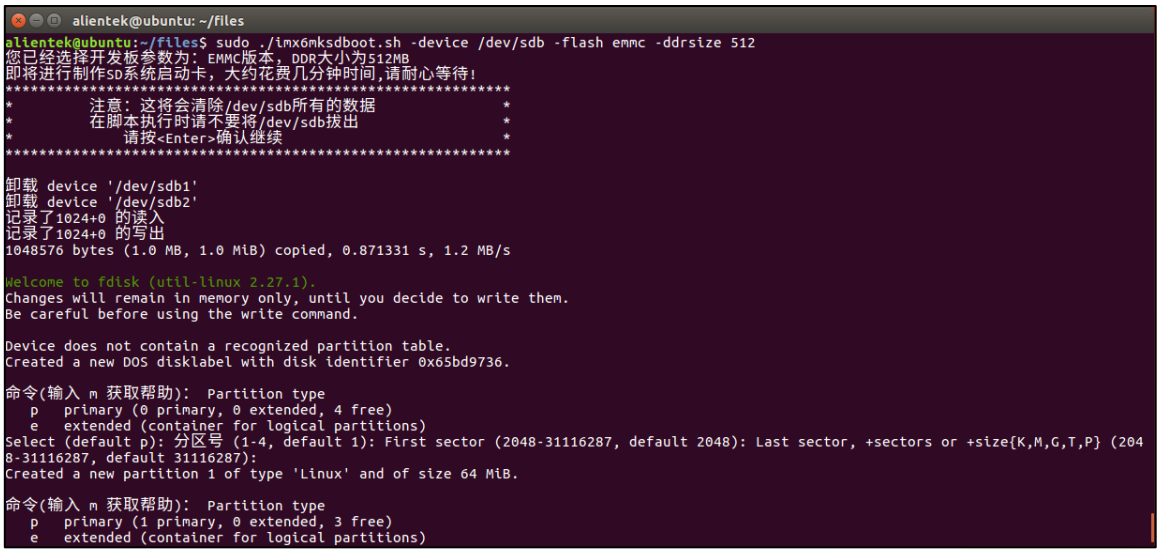


图 2.2.2.1 6 执行固化系统到 SD 卡

在固化的过程中，会卸载 TF 卡，在脚本执行时，鼠标不要离开 Ubuntu 虚拟机，否则可能在脚本卸载 TF 卡时，TF 卡连接到 PC 主机上去了，这样脚本就无法找到 TF 卡执行，就会提示“mount: special device /dev/sdb1 does not exist”这样的错误。固化时有中英文结合提示固化的过程，固化完成如下图。

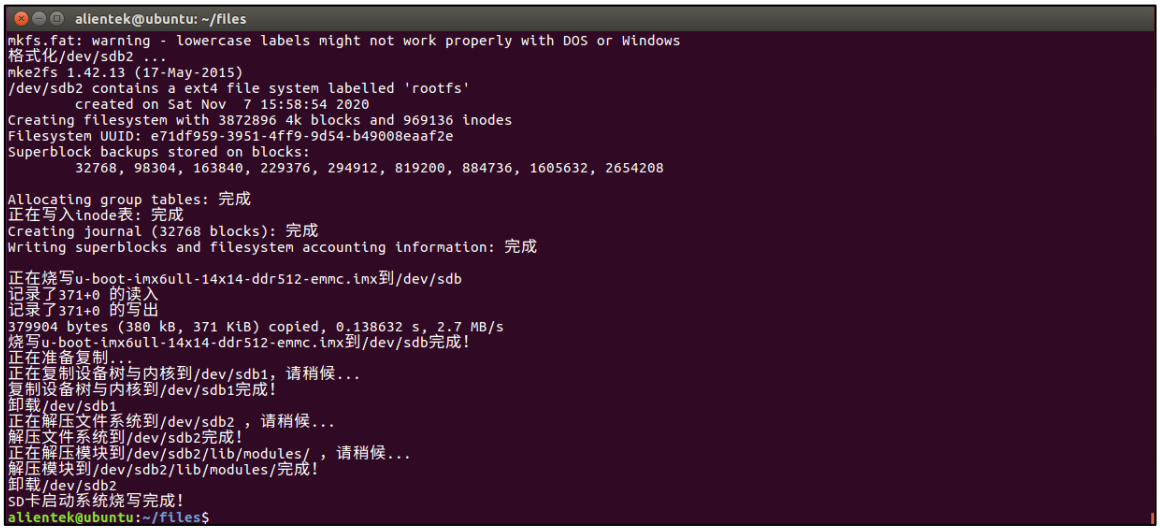


图 2.2.2.1 7 固化完成截图

按连接 TF 卡到 Ubuntu 的方法，再点击断开即可退出 TF 卡。固化完成后，将拨码开关拨至 TF 启动方式 10000010，启动系统即可。

2.2.2.2 固化系统到 eMMC

使用前提:用户核心板类型带 eMMC 存储介质，及 2.2.1.1 小节或者 2.2.2.1 小节制作好的 TF 卡启动系统。同样的，拷贝开发板光盘 A-基础资料->5、开发工具->4、正点原子 MFG_TOOL 出厂固件烧录工具->mfgtool->Profiles->Linux->OS Firmware->files 文件夹到制作好的 TF 系统启动卡里面的/home/root 目录（本文拷贝到/home/root 目录，用户可任意目录）。

```
root@ATK-IMX6U:~/files# ls
README.txt          imx6mknandboot.sh
boot                imx6mksdboot.sh
filesystem          modules
imx6mkemmcboot.sh   ''$'\345\277\205\350\257\273\350\257\264\346\230\216''.txt'
root@ATK-IMX6U:~/files#
```

图 2.2.2.2 1 拷贝 files 文件夹到开发板 SD 启动卡文件系统中

修改 eMMC 固化脚本的权限

```
chmod +x imx6mkemmcboot.sh
```

```
root@ATK-IMX6U:~/files# chmod  +x imx6mkemmcboot.sh
root@ATK-IMX6U:~/files# _
```

图 2.2.2.2 2 赋予 im6mkemmcboot.sh 可执行权限

执行./imx6mkemmcboot.sh --help 查看脚本的使用说明

```
./imx6mkemmcboot.sh --help
```

```
root@ATK-IMX6U:~/files#
root@ATK-IMX6U:~/files# ./imx6mkemmcboot.sh --help


用法: imx6mkemmcboot.sh [选项] <(必选)-device> <(可选)-ddrsize>
用法示例:
./imx6mkemmcboot.sh -device /dev/mmcblk1
./imx6mkemmcboot.sh -device /dev/mmcblk1 -ddrsize 512
命令选项:
  -device              eMMC块设备节点（一般eMMC设备节点为/dev/mmcblk1）
  -ddrsize             请选择DDR大小 （512 | 256）
可选选项:
  --version             打印版本信息.
  --help                打印帮助信息.

root@ATK-IMX6U:~/files#
```

图 2.2.2.2 3 查看 imx6mkemmcboot.sh 的脚本使用说明

使用 fdisk 指令查看 eMMC 挂载节点，一般挂载节点为/dev/mmcblk1，测试的 eMMC 为 8GB 存储容量的。可以看到下图/dev/mmcblk1 就是 eMMC 的挂载节点。

```
fdisk -l
```

```
Disk /dev/mmcblk1: 7.3 GiB, 7818182656 bytes, 15269888 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x8fa9e007

Device         Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk1p1       20480   282623   262144  128M  c W95 FAT32 (LBA)
/dev/mmcblk1p2      282624 15269887 14987264  7.2G 83 Linux
```

图 2.2.2.2 4 查看 eMMC 挂载节点

用法说明：

用法: imx6mkemmcboot.sh [选项] <(必选)-device> <(可选)-ddrsize>

（1） -device：指明设备节点（eMMC 挂载的节点如/dev/mmcblk1），必需加这个参数

（2） -ddrsize：指明核心板上的 ddr 容量大小，可选为（512|256）MB

比如现在用户是核心板的 ddr 容量大小是 512MB（原子在售的 eMMC 核心板 ddr 大小都为 512MB），eMMC 挂载节点为/dev/mmcblk1。那么固化的指令如下，执行指令后脚本会有中文再次询问 eMMC 所挂载的节点是否对应，将会清空 eMMC 上的所有数据，请注意备份重要的数据。按 Enter 键确认后继续，固化系统到 eMMC 需要大约需要几分钟时间。

```
./imx6mkemmcboot.sh -device /dev/mmcblk1 -ddrsize 512
```

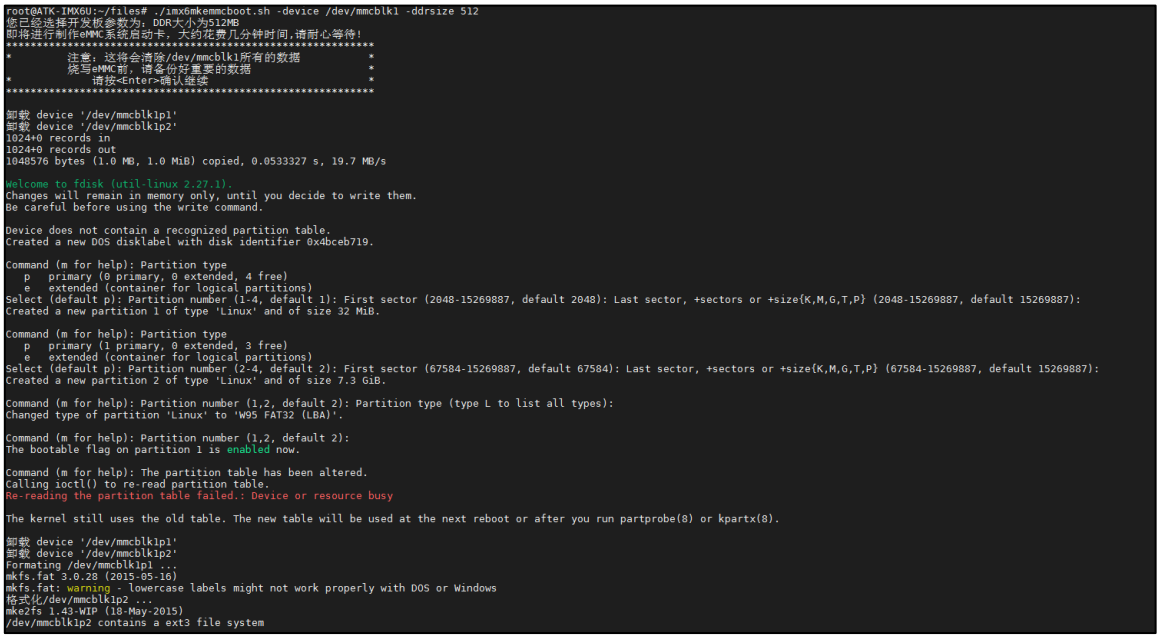


图 2.2.2.2 5 执行固化脚本指令

固化系统完成如下图。

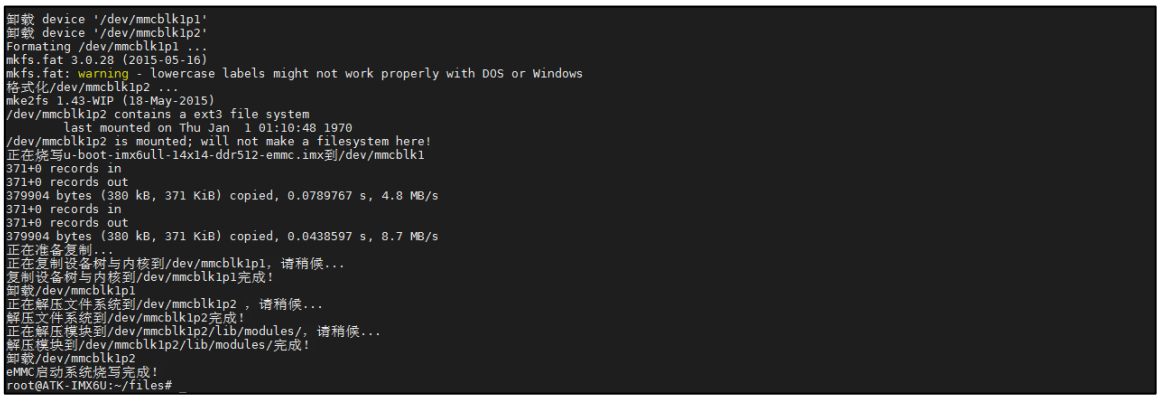


图 2.2.2.2 6 固化系统到 eMMC 完成

固化完成后，将拨码开关拨至 eMMC 启动方式 10100110，启动系统即可。

2.2.2.3 固化系统到 NAND FLASH

使用前提:用户核心板类型带 Nand Flash 存储介质，及 2.2.1.1 小节或者 2.2.2.1 小节制作好的 TF 卡启动系统。同样地，拷贝开发板光盘 A-基础资料->5、开发工具->4、正点原子 MFG_TOOL 出厂固件烧录工具->mfgtool->Profiles->Linux->OS Firmware->files 文件夹到制作好的 TF 系统启动卡里面的/home/root 目录（本文拷贝到/home/root 目录，读者可任意目录）。

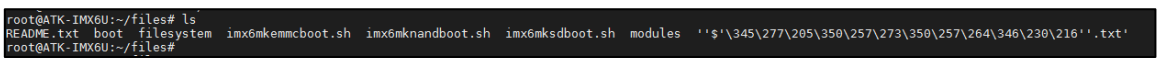


图 2.2.2.3 1 拷贝 files 文件夹到开发板 TF 启动卡文件系统中

使用 cat 指令查看 MTD 分区表，打印结果如下图表示存在 Nand Flash。

```
cat /proc/mtd
```

```
root@ATK-IMX6U:~/files# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00020000 "u-boot"
mtd1: 00020000 00020000 "env"
mtd2: 00100000 00020000 "logo"
mtd3: 00100000 00020000 "dtb"
mtd4: 00800000 00020000 "kernel"
mtd5: 1f1e0000 00020000 "rootfs"
root@ATK-IMX6U:~/files#
```

图 2.2.2.3 2 查看 Nand Flash 的分区信息

赋予 NandFlash 固化脚本的可执行权限

```
chmod +x imx6mknandboot.sh
```

```
root@ATK-IMX6U:~/files# chmod +x imx6mknandboot.sh
root@ATK-IMX6U:~/files#
```

图 2.2.2.3 3 赋予 imx6mknandboot.sh 可执行权限

执行./imx6mknandboot.sh --help 查看脚本的使用说明

```
./imx6mknandboot.sh --help
```

```
root@ATK-IMX6U:~/files# ./imx6mknandboot.sh --help


用法: imx6mknandboot.sh [选项] <(可选)-ddrsize>
用法示例:
./imx6mknandboot.sh
./imx6mknandboot.sh -ddrsize 512
命令选项:
  -ddrsize             请选择DDR大小 （512 | 256）
可选选项:
  --version             打印版本信息.
  --help                打印帮助信息.

root@ATK-IMX6U:~/files# _
```

图 2.2.2.3 4 查看 imx6mknandboot.sh 使用说明

用法说明：

用法:imx6mknandboot.sh [选项] <(可选)-ddrsize>

-ddrsize：指明核心板上的 ddr 容量大小，可选为（512|256）MB

正点原子在售的 NandFlash 类型核心板的 ddr 容量大小都是 256MB，那么固化的指令如下。这将会清空 NandFlash 上的所有数据，请注意备份重要的数据。按 Enter 键确认后继续，固化系统到 NandFlash 需要大约需要几分钟时间。

```
./imx6mknandboot.sh -ddrsize 256
```

```
root@ATK-IMX6U:~/files# ./imx6mknandboot.sh -ddrsize 256
您已经选择开发板参数为：DDR大小为256MB
即将进行固化NandFlash系统，大约花费几分钟时间,请耐心等待!
************************************************************
*        注意：这将会清除NandFlash所有的数据               *
*       在脚本执行时请勿断电或者中断烧写过程              *
*              请按<Enter>确认继续                         *
************************************************************

正在擦除Uboot分区...
Erasing 128 Kibyte @ 3e0000 -- 100 % complete
正在烧写U-boot...
Erasing 128 Kibyte @ 0 -- 100 % complete
Erasing 128 Kibyte @ e0000 -- 100 % complete
正在擦除设备树分区...
Erasing 128 Kibyte @ e0000 -- 100 % complete
正在烧写设备树...
正在擦除内核分区...
Erasing 128 Kibyte @ 7e0000 -- 100 % complete
正在烧写内核...
正在擦除文件系统分区，请稍候...
Erasing 128 Kibyte @ 1f140000 -- 99 % complete flash_erase: Skipping bad block at 1f160000
flash_erase: Skipping bad block at 1f180000
flash_erase: Skipping bad block at 1f1a0000
flash_erase: Skipping bad block at 1f1c0000
Erasing 128 Kibyte @ 1f1c0000 -- 100 % complete
ubiformat: mtd5 (nand), size 52205976 bytes (497.9 MiB), 3983 eraseblocks of 131072 bytes (128.0 KiB), min. I/O size 2048 bytes
libscan: scanning eraseblock 3982 -- 100 % complete
ubiformat: 3979 eraseblocks are supposedly empty
ubiformat: 4 bad eraseblocks found, numbers: 3979, 3980, 3981, 3982
ubi[  902.644325] ubi0: attaching mtd5-- 98 % complete
ubiformat: formatting eraseblock 3982 -- 100 % complete
```

图 2.2.2.3 5 执行固化系统到 Nand Flash 指令

固化系统完成如下图图 2.2.2.3 6。

```
您已经选择开发板参数为：DDR大小为256MB
即将进行固化NandFlash系统，大约花费几分钟时间,请耐心等待!
************************************************************
*       注意：这将会清除NandFlash所有的数据                *
*      在脚本执行时请勿断电或者中断烧写过程                *
*            请按<Enter>确认继续                           *
************************************************************

正在擦除Uboot分区...
Erasing 128 Kibyte @ 3e0000 -- 100 % complete
正在烧写U-boot...
Erasing 128 Kibyte @ 0 -- 100 % complete
Erasing 128 Kibyte @ e0000 -- 100 % complete
正在擦除设备树分区...
Erasing 128 Kibyte @ e0000 -- 100 % complete
正在烧写设备树...
正在擦除内核分区...
Erasing 128 Kibyte @ 7e0000 -- 100 % complete
正在烧写内核...
正在擦除文件系统分区，请稍候...
Erasing 128 Kibyte @ 1f140000 -- 99 % complete flash_erase: Skipping bad block at 1f160000
flash_erase: Skipping bad block at 1f180000
flash_erase: Skipping bad block at 1f1a0000
flash_erase: Skipping bad block at 1f1c0000
Erasing 128 Kibyte @ 1f1c0000 -- 100 % complete
ubiformat: mtd5 (nand), size 522059776 bytes (497.9 MiB), 3983 eraseblocks of 131072 bytes (128.0 KiB), min. I/O size 2048 bytes
libscan: scanning eraseblock 3982 -- 100 % complete
ubiformat: 3979 eraseblocks are supposedly empty
ubiformat: 4 bad eraseblocks found, numbers: 3979, 3980, 3981, 3982
ubi[  902.644325] ubi0: attaching mtd5-- 98 % complete
ubiformat: formatting eraseblock 3982 -- 100 % complete
[  903.699487] ubi0: scanning is finished
[  903.719541] ubi0: attached mtd5 (name "rootfs", size 497 MiB)
[  903.725461] ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
[  903.732350] ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
[  903.740149] ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
[  903.747301] ubi0: good PEBs: 3979, bad PEBs: 4, corrupted PEBs: 0
[  903.754160] ubi0: user volume: 0, internal volumes: 1, max. volumes count: 128
[  903.761408] ubi0: max/mean erase counter: 0/0, WL threshold: 4096, image sequence number: 1603539058
[  903.771222] ubi0: available PEBs: 3899, total reserved PEBs: 80, PEBs reserved for bad PEB handling: 76
[  903.780726] ubi0: background thread "ubi_bgt0d" started, PID 1030
UBI device number 0, total 3979 LEBs (505237504 bytes, 481.8 MiB), available 3899 LEBs (495079424 bytes, 472.1 MiB), LEB size 126976 bytes (124.0 KiB)
Set volume size to 495079424
Volume ID 0, size 3899 LEBs (495079424 bytes, 472.1 MiB), LEB size 126976 bytes (124.0 KiB), dynamic, name "rootfs", alignment 1
[  903.892102] UBIFS (ubi0:0): default file-system created
[  903.898044] UBIFS (ubi0:0): background thread "ubifs_bgt0_0" started, PID 1034
[  903.938433] UBIFS (ubi0:0): UBIFS: mounted UBI device 0, volume 0, name "rootfs"
[  903.945946] UBIFS (ubi0:0): LEB size: 126976 bytes (124 KiB), min./max. I/O unit sizes: 2048 bytes/2048 bytes
[  903.956747] UBIFS (ubi0:0): FS size: 493047808 bytes (470 MiB, 3883 LEBs), journal size 24633344 bytes (23 MiB, 194 LEBs)
[  903.967807] UBIFS (ubi0:0): reserved for root: 4952683 bytes (4836 KiB)
[  903.975107] UBIFS (ubi0:0): media format: w4/r0 (latest is w4/r0), UUID ABC69936-7391-4D3B-A5A5-08F935D3C255, small LPT model
正在解压文件系统到mtd5分区，请稍候...
正在解压模块到mtd5分区，请稍候...
[ 1180.272009] UBIFS (ubi0:0): un-mount UBI device 0
[ 1180.277068] UBIFS (ubi0:0): background thread "ubifs_bgt0_0" stops
[ 1180.341701] ubi0: detaching mtd5
[ 1180.351816] ubi0: mtd5 is detached
NandFlash启动系统烧写完成!
root@ATK-IMX6U:~/files#
```

图 2.2.2.3 6 固化完成

固化完成后，将拨码开关拨至 Nand Flash 启动方式 10001001，启动系统即可。

2.3 登录开发板

2.3.1 拔码开关设置

请根据个人核心板上的存储介质类型，选择不同的方式启动动系统。如下图参照底板原理图说明：

SW								
D1	D2	D3	D4	D5	D6	D7	D8	BOOT DEVICE
OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	USB
ON	OFF	OFF	OFF	OFF	OFF	ON	OFF	MicroSD
ON	OFF	ON	OFF	OFF	ON	ON	OFF	EMMC
ON	OFF	OFF	OFF	ON	OFF	OFF	ON	NAND

图 2.3.1 1 底板原理图拨码开关设置

解释：OFF 为 0，ON 为 1。

（1） USB OTG 烧写设置：0100 0000

（2） SD 卡启动设置：1000 0010

（3） EMMC 启动设置：1010 0110

（4） NAND FLASH 启动设置：1000 1001

用户选择正确的启动方式后，开发板上电，进入系统后，开发板自动登录，无需输入用户名及密码，文件系统已经设置 root 用户自动登录。

第三章 ATK I.MX6U 功能测试

本章简介如下。

- 测试板子的资源，验证硬件功能。在这里我们能学习到 Linux 开发板的测试方法。对我们学习开发板有很大作用。

3.1. LED 与蜂鸣器测试

说明：用户 LED 一个，蜂鸣器一个。开发板启动时，DS0 作为心跳灯，用于指示系统的运行

开发板与 LED 和蜂鸣器对应的管脚关系如下：

开发板	GPIO03	SNVS_TAMPER1
ALPHA/Mini	DS0	BEEP

进入开发板系统，在串口终端执行指令控制对应的 IO 来控制对应的器件：

开发板上启动后 DS0 默认是[heartbeat]模式，执行如下指令改变当前触发模式，改成[none]模式就可以通过指令来控制 LED 的亮灭了。

```
echo none > /sys/class/leds/sys-led/trigger          // 改变 LED 的触发模式
echo 1 > /sys/class/leds/sys-led/brightness          // 点亮 LED
echo 0 > /sys/class/leds/sys-led/brightness          // 熄灭 LED
```

```
root@ATK-IMX6U:~# echo none > /sys/class/leds/sys-led/trigger
root@ATK-IMX6U:~# echo 1 > /sys/class/leds/sys-led/brightness
root@ATK-IMX6U:~# echo 0 > /sys/class/leds/sys-led/brightness
root@ATK-IMX6U:~#
```

图 3.1 1 输入指令控制 led

因为底板的蜂鸣器用配置成了 gpio-leds 模式，同理蜂鸣器也可以用这样的指令来控制，默认 BEEP 的触发模式为[none]。

```
echo 1 > /sys/class/leds/beep/brightness          // 鸣叫
echo 0 > /sys/class/leds/beep/brightness          // 关闭
```

```
root@ATK-IMX6U:~# echo 1 > /sys/class/leds/beep/brightness
root@ATK-IMX6U:~# echo 0 > /sys/class/leds/beep/brightness
root@ATK-IMX6U:~# _
```

图 3.1 2 输入指令控制 beep

3.2 按键测试

底板上按键对应的管脚关系如下：

开发板	GPIO18
ALPHA/Mini	KEY0

进入开发板系统，在串口终端执行如下指令查看按键所对应的输入事件

```
lsinput
```

```
root@ATK-IMX6U:~# lsinput
/dev/input/event0
   bustype : BUS_HOST
   vendor  : 0x0
   product : 0x0
   version : 0
   name    : "20cc000.snvs:snvs-powerkey"
   phys    : "snvs-pwrkey/input0"
   bits ev : EV_SYN EV_KEY

/dev/input/event1
   bustype : BUS_I2C
   vendor  : 0x0
   product : 0x0
   version : 0
   name    : "EP0820M09"
   bits ev : EV_SYN EV_KEY EV_ABS

/dev/input/event2
   bustype : BUS_HOST
   vendor  : 0x1
   product : 0x1
   version : 256
   name    : "gpio_keys@0"
   phys    : "gpio-keys/input0"
   bits ev : EV_SYN EV_KEY EV_REP

open /dev/input/event3: No such file or directory
root@ATK-IMX6U:~#
```

事件节点

事件类型

图 3.2 1 查看按键的输入事件

可以从上图看出按键事件号为 event2，触摸屏占用了 event1，所以当触摸屏没有插上时按

键事件号为不一定为 event2！执行下面的指令，进行按键测试，按下底板上的 KEY0，打印出按键输入事件的信息如下，按“Ctrl + c”终止指令。

指令提示：hexdump 或者 od -x 指令都是以十六进制的形式打印出输入事件信息。由于文件系统没提供 hexpdump 指令，所以只测试 od 指令。

```
od -x /dev/input/event2
```

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# od -x /dev/input/event2
0000000 da05 5fa6 f915 000c 0014 0000 0190 0000
0000020 da05 5fa6 f915 000c 0014 0001 0050 0000
0000040 da05 5fa6 f915 000c 0001 0072 0001 0000
0000060 da05 5fa6 f915 000c 0000 0000 0000 0000
0000100 da06 5fa6 4edc 0000 0001 0072 0000 0000
0000120 da06 5fa6 4edc 0000 0000 0000 0000 0000
0000140 da06 5fa6 60e6 000a 0001 0072 0001 0000
0000160 da06 5fa6 60e6 000a 0000 0000 0000 0000
0000200 da06 5fa6 2009 000d 0001 0072 0000 0000
0000220 da06 5fa6 2009 000d 0000 0000 0000 0000
^C
root@ATK-IMX6U:~#
```

图 3.2 2 打印按键输入事件的信息

3.3 LCD 触摸屏

ALPHA/Mini 开发板已经兼容所有本公司的 RGB LCD 屏幕，由于屏幕存在 id 的关系，开发板上电自动识别屏幕，进行 LCD 触摸屏实验时请先断电插上 RGB 屏幕，再启动系统。

屏幕尺寸	触摸芯片	屏幕 ID
4.3 寸屏（480x272）	gt9xx	0x00
4.3 寸屏（800x480）	gt9xx/gt1151	0x04
7 寸屏（800x480）	ft5x06	0x01
7 寸屏（1024x600）	ft5x06/cst340(cst340 兼容 ft5x06 驱动)	0x02
10.1 寸屏（1280x800）	gt9xx	0x03

3.3.1 LCD 背光调节

LCD 屏幕的背光支持 8 级变化，亮度级数为 0～7，默认为 7。

查看 LCD 屏幕最大亮度等级

```
cat /sys/devices/platform/backlight/backlight/backlight/max_brightness
```

```
root@ATK-IMX6U:~# cat /sys/devices/platform/backlight/backlight/backlight/max_brightness
7
root@ATK-IMX6U:~#
```

图 3.3.1 1 查看 LCD 屏幕背光最大亮度等级

查看当前 LCD 屏幕背光亮度等级

```
cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness
```

```
root@ATK-IMX6U:~# cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness
7
root@ATK-IMX6U:~#
```

图 3.3.1 2 查看当前 LCD 屏幕背光亮度等级

修改当前 LCD 屏幕背光亮度等级，修改后再查看当前亮度等级

```
echo 5 > /sys/class/backlight/backlight/brightness // 修改亮度等级为 5
cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness
```

```
root@ATK-IMX6U:~# echo 5 > /sys/class/backlight/backlight/brightness
root@ATK-IMX6U:~# cat /sys/devices/platform/backlight/backlight/backlight/actual_brightness
5
root@ATK-IMX6U:~# _
```

图 3.3.1 3 查看修改后的 LCD 屏幕背光亮度等级

3.3.2 LCD 触摸测试

先退出 Qt 桌面，直接点击桌面的“设置”，点击设置项里的退出桌面即可！如果没有这项，用户的出厂系统还不是最新的，请使用正点原子 ALPHA 最新的资料，参考第 2.2 小节的更新系统即可！

以下实验是对本公司 7 寸屏 800*480 分辨率的屏幕进行测试。

```
ts_test                                    //可按“Ctrl + c”停止
```

```
root@ATK-IMX6U:~# ts_test
1604770876.385795:    121    233    255
1604770876.394710:    121    233    255
1604770876.398531:    121    233    255
1604770876.405040:    121    233    255
1604770876.408610:    121    233    255
1604770876.417550:    121    233    255
1604770876.428003:    121    233    255
1604770876.438423:    121    233    255
1604770876.448921:    121    233    255
1604770876.459377:    121    233    255
1604770876.469787:    121    233    255
1604770876.480937:    122    233    255
1604770876.491294:    126    235    255
1604770876.501885:    130    236    255
1604770876.512356:    136    238    255
1604770876.522690:    143    241      0
^Csignal 2 caught
root@ATK-IMX6U:~#
```

图 3.3.2 1 ts_test 的测试结果

LCD 测试程序如下图。点击 Draw，就可以开始画图，7 寸屏 800*480 分辨率效果。

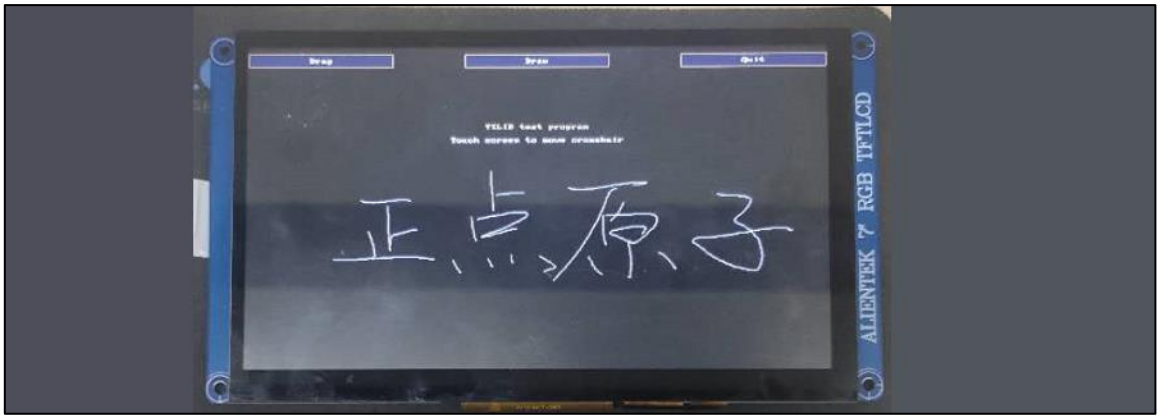


图 3.3.2 2 ts_test 测试触摸情况

关闭桌面程序后，可执行下面的指令进行重启桌面程序

```
/opt/QDesktop > /dev/null 2>&1 &
```

注意：这里不需要使用 ts_calibrate 校准。ts_calibrate 一般是针对电阻屏校准的。使用 ts_calibrate 后会在/etc 目录下生成一个坐标参考文件 pointercal。如果您的/etc 目录下有 pointercal 这个坐标参考文件，请把它删除。否则可能影响触摸坐标上报的数据值。

3.3.3 LCD 显示控制

进入/退出睡眠模式，注：这里指控制 LCD 进入睡眠模式（实质是操控 fb0），并不是整机进入睡眠模式。

进入睡眠/熄屏模式

```
echo "4" > /sys/class/graphics/fb0/blank
```

```
root@ATK-IMX6U:~# echo "4" > /sys/class/graphics/fb0/blank
root@ATK-IMX6U:~#
```

图 3.3.3 1 关闭屏幕背光

退出睡眠/亮屏模式

```
echo "0" > /sys/class/graphics/fb0/blank
```

```
root@ATK-IMX6U:~# echo "0" > /sys/class/graphics/fb0/blank
root@ATK-IMX6U:~#
```

图 3.3.3 2 开启屏幕背光

3.4 串口测试

3.4.1 RS232 串口测试

ALPHA	MINI
本实验支持	本实验不支持，但是可以使用 USB 转换器模块的 TTL UART 直接接 MINI 板 JP3 处测试

测试前准备正点原子 USB 转换器和 RS232 转接头（或者直接用 USB 转公头 RS232 串口线）。如下图，下图准备的是正点原子的 USB 转换器。可以测试 RS232 与 RS485 等。（注：3.4.2 小节测试 RS485 也是用这个模块。）

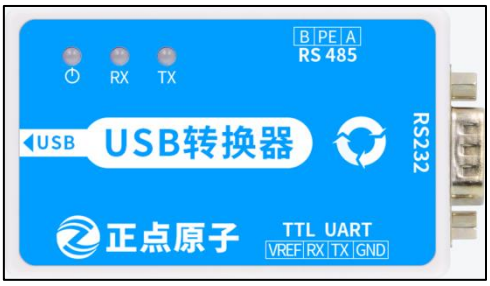


图 3.4.1 1 正点原子 USB 转换器模块

底板 RS232 接口与 RS485 接口的 Tx 与 Rx 是共用的，使用时需要使用跳线帽进行切换。RS232 串口线的一头公头接到开发板底板的 COM3 接口处，另一头母头连接 USB 转换器再连接计算机的 USB 接口。测试前请将跳帽将 U3_Tx 与 COM3_Rx 连接，U3_Rx 与 COM3_Tx 连接。连接方法示意及切换跳线帽的位置如下图图 3.4.1. 2。

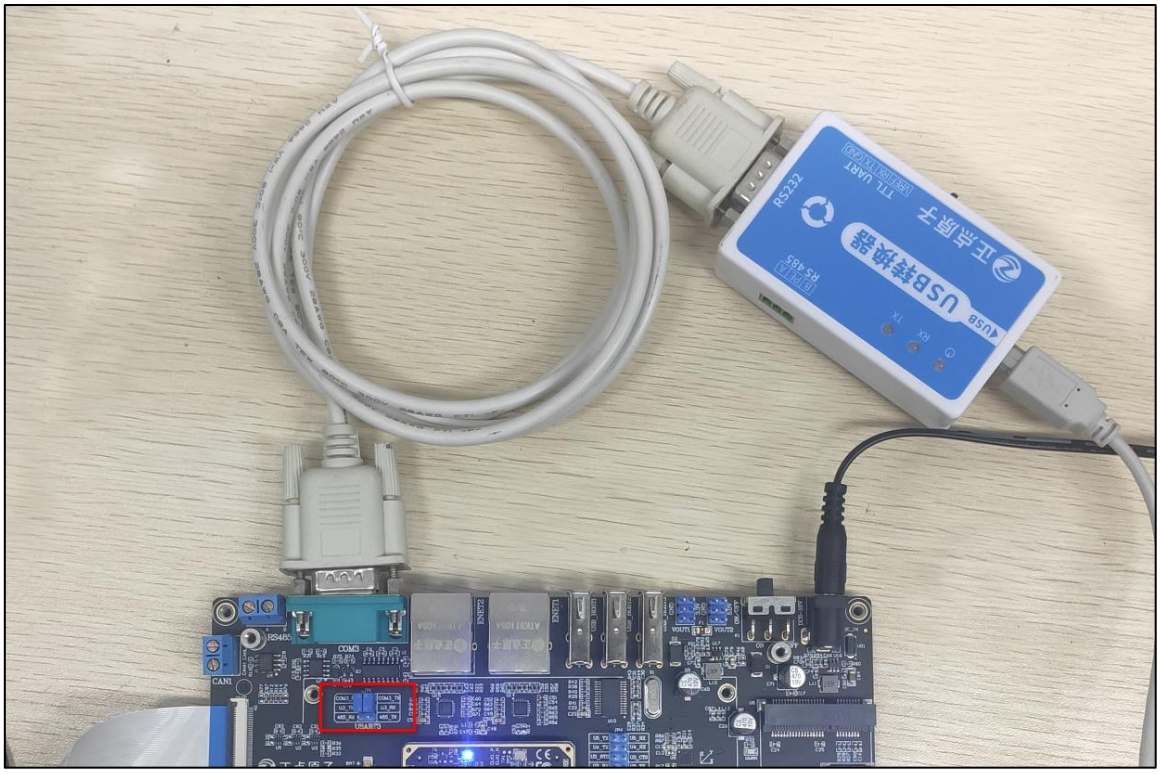


图 3.4.1 2 RS232 接口与切换跳线帽的位置

在计算机的设备查看 USB 转换器的端口号。编者的端口号有两个，一个是开发板 USB 调试串口的，另一个就是 USB 转换器的端口号了。

图 3.4.1 3 在设备管理器查看 USB 转换器的端口号

在 MobaXterm 里连接 COM91，和调试串口设置的方法一样，设置波特率为 115200，8N1。可以看到 COM90 为编者的调试串口（USART1），COM91 为开发板底板的 COM3(USART3)串口。

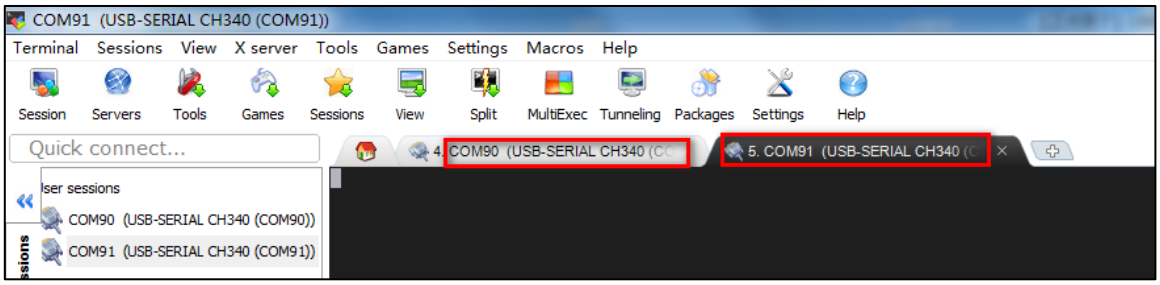


图 3.4.1 4 连接的 RS232 串口 COM3

在 USART1 串口调试终端 COM90 输入下面的指令，将 RS232 也设置成一个串口终端。

```
setsid getty 115200 /dev/ttymxc2
```

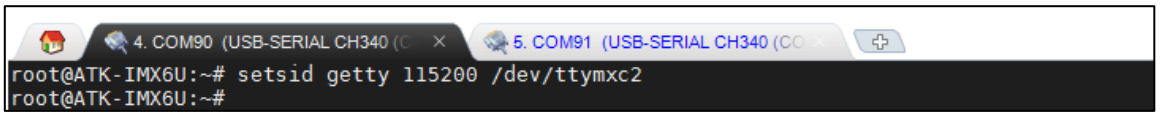


图 3.4.1 5 使用指令设置 RS232 为一个串口终端

这样可以像调试串口一样输入登录用户名 root，即可进入系统。这样能输入指令并返回结果，表明 RS232 串口正常。

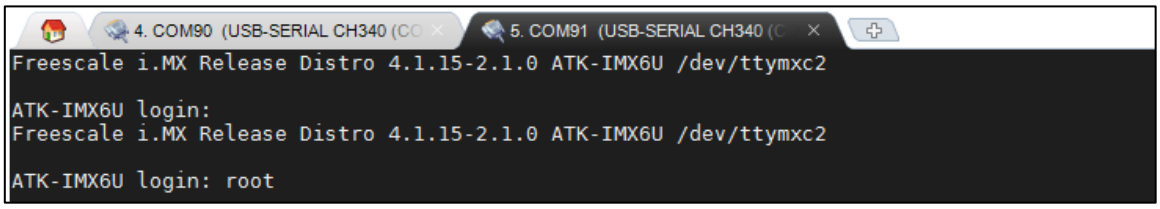


图 3.4.1 6 输入“root”指令登录开发板

3.4.2 RS485 串口测试

与 RS232 测试方法一样，使用 USB 转换器测试 RS485 接口（或者用户手上有其他 RS232 转 RS485 的模块亦可）。测试前请将跳帽将 U3_Tx 与 485_Rx 连接，U3_Rx 与 485_Tx 连接。切换跳线帽的位置如下图。

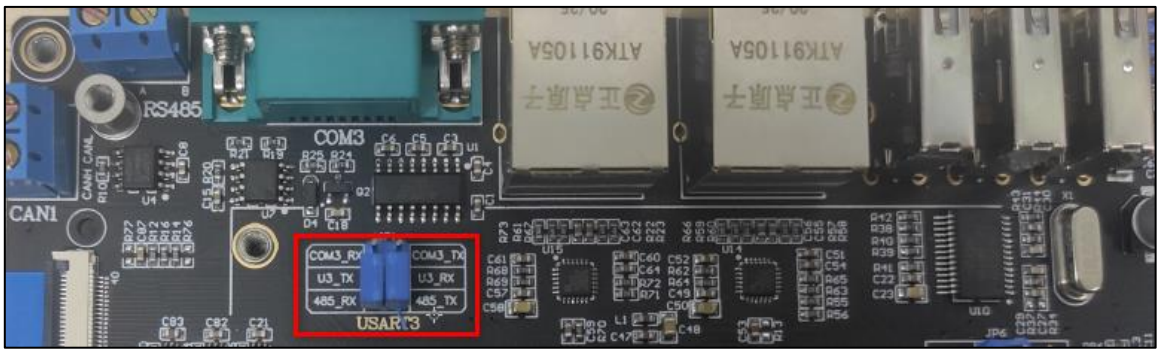


图 3.4.2 1 切换跳线帽的位置

将 USB 转换器上的 RS485 端口的 A 用铜线接到开发板 RA485 端口的 A 处，USB 转换器上的 RS485 端口的 B 用铜线接到开发板 RA485 端口的 B 处。

在计算机的设备查看 USB 转换器的端口号。编者的端口号有两个，一个是开发板 USB 调试串口的，另一个就是 USB 转换器的端口号了。

剩余测试步骤和前小节 3.4.1 的一样，不再重复累赘了。

3.5 DDR 测试

Memtester 简单介绍

Memtester 主要是捕获内存错误和一直处于很高或者很低的坏位， 其测试的主要项目有随机值，异或比较，减法，乘法，除法，与或运算等等。 通过给定测试内存的大小和次数，可以对系统现有的内存进行上面项目的测试。

memtester [-p PHYSADDR] <MEMORY> [ITERATIONS]

参数说明：

- MEMORY 申请测试内存的数量，单位默认是 megabytes(兆)，也可以是 B K M G。
- ITERATIONS 测试的次数，默认是无限

使用文件系统自带的 Memtester 测试工具申请 8MB 内存数量测试做 1 次 DDR 测试。执行如下指令。

```
memtester 8M 1
```

```
root@ATK-IMX6U:~# memtester 8M 1
memtester version 4.3.0 (32-bit)
Copyright (C) 2001-2012 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffff000
want 8MB (8388608 bytes)
got  8MB (8388608 bytes), trying mlock ...locked.
Loop 1/1:
  Stuck Address       : ok
  Random Value        : ok
  Compare XOR         : ok
  Compare SUB         : ok
  Compare MUL         : ok
  Compare DIV         : ok
  Compare OR          : ok
  Compare AND         : ok
  Sequential Increment: ok
  Solid Bits          : ok
  Block Sequential    : ok
  Checkerboard        : ok
  Bit Spread          : ok
  Bit Flip            : ok
  Walking Ones        : ok
  Walking Zeroes      : ok

Done.
root@ATK-IMX6U:~#
```

图 3.5 1 测试 DDR

3.6 TF(SD)卡读写测试

本实验测试 SD 卡读写速度，建议使用 SD 卡作为系统启动卡。

指令提示：

time 命令常用于测量一个命令的运行时间，dd 用于复制，从 if(input file)文件读出，写到 of(output file)指定的文件，bs 是每次写块的大小，count 是读写块的数量。"if=/dev/zero"不产生 IO，即可以不断输出数据，因此可以用来测试纯写速度。

3.6.1 TF(SD)卡写速度测试

本次对 TF 系统卡的第一个分区写 50MiB 数据（注意这里读写数据越大测试出来计算出来的结果越平均与接近实际值，但要注意 TF 卡第一个分区的实际大小）。执行下面的指令测试 TF 卡的写速度。

```
time dd if=/dev/zero of=/run/media/mmcblk0p1/test bs=1024k count=50 conv=fdatasync
```

```
root@ATK-IMX6U:~# time dd if=/dev/zero of=/run/media/mmcblk0p1/test bs=1024k count=50 conv=fdatasync
50+0 records in
50+0 records out
52428800 bytes (52 MB, 50 MiB) copied, 6.58148 s, 8.0 MB/s

real    0m6.589s
user    0m0.000s
sys     0m1.590s
root@ATK-IMX6U:~# _
```

图 3.6.1 1 执行指令测试 SD 卡的写速度

这里一共写入 50 MiB test 文件，速度为 8.0 MB/s。

3.6.2 TF(SD)卡读速度测试

小提示：

因为 LINUX 的内核机制，一般情况下不需要特意去释放已经使用的 cache。这些 cache 内容可以增加文件以及的读写速度。

执行下面指令清除缓存

```
echo 3 > /proc/sys/vm/drop_caches
```

```
root@ATK-IMX6U:~# echo 3 > /proc/sys/vm/drop_caches
[  380.393742] sh (646): drop_caches: 3
root@ATK-IMX6U:~# _
```

图 3.6.2 1 执行指令清除缓存

执行下面的指令读取前面用 dd 指令写入的 test 文件

```
time dd if=/run/media/mmcblk0p1/test of=/dev/null bs=1024k
```

```
root@ATK-IMX6U:~# time dd if=/run/media/mmcblk0p1/test of=/dev/null bs=1024k
50+0 records in
50+0 records out
52428800 bytes (52 MB, 50 MiB) copied, 3.14981 s, 16.6 MB/s

real    0m3.187s
user    0m0.000s
sys     0m0.460s
root@ATK-IMX6U:~#
```

图 3.6.2 2 读取 test 文件

这里一共读出了 50 MiB test 文件，速度为 16.6MB/s，测试完成后，可以把 test 文件使用 rm 指令删除。

3.7 NAND FLASH 读写速度测试

本实验测试要求用 SD 卡启动卡启动，使用 NAND FLASH 版本的核心板进行测试。实际上读写数据量越大，数据越平均，越接近实际值。

```
cat /proc/mtd
```

```
root@ATK-IMX6U:~# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00400000 00020000 "u-boot"
mtd1: 00020000 00020000 "env"
mtd2: 00100000 00020000 "logo"
mtd3: 00100000 00020000 "dtb"
mtd4: 00800000 00020000 "kernel"
mtd5: 1f1e0000 00020000 "rootfs"
root@ATK-IMX6U:~#
```

图 3.7 1 查看 Nand Flash 的分区

写数据前需要对操作的分区进行擦除，注意该操作会清除该操作分区的数据，请提前做好数据备份。

```
flash_erase /dev/mtd5 0 0
```

```
root@ATK-IMX6U:~# flash_erase /dev/mtd5 0 0
Erasing 128 Kibyte @ 1f140000 -- 99 % complete flash_erase: Skipping bad block at 1f160000
flash_erase: Skipping bad block at 1f180000
flash_erase: Skipping bad block at 1f1a0000
flash_erase: Skipping bad block at 1f1c0000
Erasing 128 Kibyte @ 1f1c0000 -- 100 % complete
root@ATK-IMX6U:~#
```

图 3.7 2 擦除 mtd5 文件系统分区

执行如下指令往 NAND FLASH 的 mtd5 文件系统分区写入 50MiB 数据，bs 大小为 1024KB

```
time dd if=/dev/zero of=/dev/mtd5 bs=1024k count=50
```

```
root@ATK-IMX6U:~# time dd if=/dev/zero of=/dev/mtd5 bs=1024k count=50
50+0 records in
50+0 records out
52428800 bytes (52 MB, 50 MiB) copied, 8.25863 s, 6.3 MB/s

real    0m8.265s
user    0m0.000s
sys     0m5.880s
root@ATK-IMX6U:~#
```

图 3.7 3 向文件系统分区写入 50MiB 数据

执行如下指令从 NAND FLASH 的 mtd4 内核分区（8MiB）读取 8MiB 数据。

```
time dd if=/dev/mtd4 of=/dev/null bs=1024k
```

```
root@ATK-IMX6U:~# time dd if=/dev/mtd4 of=/dev/null bs=1024k
8+0 records in
8+0 records out
8388608 bytes (8.4 MB, 8.0 MiB) copied, 0.515735 s, 16.3 MB/s

real    0m0.522s
user    0m0.000s
sys     0m0.220s
root@ATK-IMX6U:~# _
```

图 3.7 4 读取内核分区数据

3.8 系统时钟与 RTC 时钟

小提示：

测试 RTC 时钟要安装上钮扣电池。

Linux 系统分两个时钟，一个是 system time（软件时钟），一个是 hardware clock（硬件时钟，6ULL 芯片内部有 RTC 硬件时钟（可能不太精准））。使用 date 和 hwclock 命令可分别查看和设定系统时间和硬件时间。系统时钟掉电即会消失，RTC 时钟在有电池的情况下会长期运行。系统时钟会在系统重启时与 RTC 时钟同步。

查看系统时钟，使用指令 date。

```
date
```

```
root@ATK-IMX6U:~# date
Sat Nov  7 18:57:17 UTC 2020
root@ATK-IMX6U:~#
```

图 3.8 1 查看系统时钟的时间

查看硬件（RTC）时钟，使用指令 hwclock。

```
hwclock
```

```
root@ATK-IMX6U:~# hwclock
Tue Sep 15 17:51:28 2020  0.000000 seconds
root@ATK-IMX6U:~#
```

图 3.8 2 查看硬件时钟

设置系统时钟，设置当前时间，然后查看设置的系统时间。例如当前时间是 2020 年 11 月 15 日 下午 20:00:00。指令如下：

```
date -s "2020-11-15 20:00:00"          // 设置当前系统时钟
date                                   // 查看当前系统时钟
```

```
root@ATK-IMX6U:~# date -s "2020-11-15 20:00:00"
Sun Nov 15 20:00:00 UTC 2020
root@ATK-IMX6U:~# date
Sun Nov 15 20:00:07 UTC 2020
root@ATK-IMX6U:~#
```

图 3.8 3 设置系统时钟

将系统时钟写入硬件时钟。

```
hwclock -w                             // 将系统时钟同步至硬件时钟
```

```
hwclock                                    // 查看硬件时钟
```

```
root@ATK-IMX6U:~# hwclock -w
root@ATK-IMX6U:~# hwclock
Sun Nov 15 20:00:51 2020  0.000000 seconds
root@ATK-IMX6U:~#
```

图 3.8.4 将系统时钟写入硬件时钟

3.9 查看系统信息

显示操作系统的内核版本号。

```
uname -a
```

```
root@ATK-IMX6U:~# uname -a
Linux ATK-IMX6U 4.1.15-g45a21e3 #1 SMP PREEMPT Wed Oct 28 10:36:39 CST 2020 armv7l armv7l armv7l GNU/Linux
root@ATK-IMX6U:~# _
```

图 3.9.1 显示内核版本号

查看系统主机名。

```
cat /etc/hostname
```

```
root@ATK-IMX6U:~# cat /etc/hostname
ATK-IMX6U
root@ATK-IMX6U:~#
```

图 3.9.2 查看系统主机名

查看系统登录开机信息，（备注：非自动登录时会打印开机信息）。

```
cat /etc/issue
```

```
root@ATK-IMX6U:~# cat /etc/issue
Freescale i.MX Release Distro 4.1.15-2.1.0 \n \l

root@ATK-IMX6U:~#
```

图 3.9.3 查看系统登录欢迎信息

查看 CPU 相关信息。

```
cat /proc/cpuinfo
```

```
root@ATK-IMX6U:~# cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 12.00
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

Hardware        : Freescale i.MX6 Ultralite (Device Tree)
Revision        : 0000
Serial          : 0000000000000000
root@ATK-IMX6U:~#
```

图 3.9.4 查看 CPU 相关信息

查看内存相关信息。

```
cat /proc/meminfo
```

```
root@ATK-IMX6U:~# cat /proc/meminfo
MemTotal:         506876 kB
MemFree:          370160 kB
MemAvailable:     361652 kB
Buffers:            9728 kB
Cached:            45564 kB
SwapCached:            0 kB
Active:            48532 kB
Inactive:          36448 kB
Active(anon):      29868 kB
Inactive(anon):      136 kB
Active(file):      18664 kB
Inactive(file):    36312 kB
Unevictable:           0 kB
Mlocked:               0 kB
HighTotal:             0 kB
HighFree:              0 kB
LowTotal:         506876 kB
LowFree:          370160 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:         29684 kB
Mapped:            27352 kB
Shmem:               320 kB
Slab:              10048 kB
SReclaimable:       3520 kB
SUnreclaim:         6528 kB
KernelStack:         576 kB
PageTables:          644 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:      253436 kB
Committed_AS:      68740 kB
VmallocTotal:    1548288 kB
VmallocUsed:        3832 kB
VmallocChunk:    1367964 kB
CmaTotal:         131072 kB
CmaFree:           97244 kB
root@ATK-IMX6U:~#
```

eMMC核心板内存约为512MB
Nand Flash核心板内存约256MB

图 3.9 5 查看内存相关信息

3.10 温度传感器

I.MX6U 芯片内部内置有温度传感器，可以实时反映 I.MX6U CPU 的内部温度

cat /sys/class/thermal/thermal_zone0/temp

```
root@ATK-IMX6U:~# cat /sys/class/thermal/thermal_zone0/temp
51440
root@ATK-IMX6U:~#
```

图 3.10 1 查看芯片内部温度

温度值即为 51.440℃（51440/1000）。

3.11 网口测试

ALPHA	MINI
本实验支持	只支持一个网口 eth0

小提示：

ALPHA 开发板有 eth0 和 eth1 两路百兆网卡。eth0 对应底板上 ENET2，eth1 对应底板上的 ENET1。可使用 ifconfig 指令来显示或者配置网络

查看网络信息。

ifconfig

```
root@ATK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 62:2e:41:62:11:3e
          inet addr:192.168.1.244  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::602e:41ff:fe62:113e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7596 errors:0 dropped:108 overruns:0 frame:0
          TX packets:128 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:732654 (715.4 KiB)  TX bytes:18303 (17.8 KiB)

eth1      Link encap:Ethernet  HWaddr fa:9c:b4:8e:6f:0d
          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)

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:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2381 (2.3 KiB)  TX bytes:2381 (2.3 KiB)

root@ATK-IMX6U:~#
```

图 3.11 1 使用 ifconfig 指令查看网卡信息

插上网线到 ENET2 处可以看到如下信息，系统自动获取了 ip，eth1 同理。

```
root@ATK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 62:2e:41:62:11:3e
          inet addr:192.168.1.244  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::602e:41ff:fe62:113e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7596 errors:0 dropped:108 overruns:0 frame:0
          TX packets:128 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:732654 (715.4 KiB)  TX bytes:18303 (17.8 KiB)

eth1      Link encap:Ethernet  HWaddr fa:9c:b4:8e:6f:0d
          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)

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:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2381 (2.3 KiB)  TX bytes:2381 (2.3 KiB)

root@ATK-IMX6U:~#
```

图 3.11 2 查看自动获取的 ip

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。

```
udhcpc -i eth0
```

```
root@ATK-IMX6U:~# udhcpc -i eth0
udhcpc (v1.24.1) started
Sending discover...
Sending select for 192.168.1.244...
Lease of 192.168.1.244 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 192.168.1.1
root@ATK-IMX6U:~#
```

图 3.11 3 使用 udhcpc 手动获取 ip

关闭与打开网口。

```
ifconfig eth1 down      // 关闭网口，网卡名字请根据实际情况修改，down 表示关闭
ifconfig eth1 up        // 打开网口，网卡名字请根据实际情况修改，up 表示打开
```

```
root@ATK-IMX6U:~# ifconfig eth1 down
root@ATK-IMX6U:~# ifconfig eth1 up
[ 1535.843305] fec 2188000.ethernet eth1: Freescale FEC PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=20b4000.ethernet:00, irq=-1)
[ 1535.857081] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
root@ATK-IMX6U:~#
```

所用网络芯片驱动名称

图 3.11 4 关闭与打开网口

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。

```
ping www.baidu.com -I eth0
```

```
root@ATK-IMX6U:~# ping www.baidu.com -I eth0
PING www.baidu.com (14.215.177.39) from 192.168.1.244 eth0: 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=55 time=6.79 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=55 time=6.08 ms
64 bytes from 14.215.177.39: icmp_seq=3 ttl=55 time=6.22 ms
64 bytes from 14.215.177.39: icmp_seq=4 ttl=55 time=5.97 ms
64 bytes from 14.215.177.39: icmp_seq=5 ttl=55 time=6.52 ms
^C
--- www.baidu.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 5.974/6.318/6.793/0.312 ms
root@ATK-IMX6U:~#
```

图 3.11 5 测试 eth0 联网

```
ping www.baidu.com -I eth1
```

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# ping www.baidu.com -I eth1
PING www.baidu.com (14.215.177.38) from 192.168.1.245 eth1: 56(84) bytes of data.
64 bytes from 14.215.177.38: icmp_seq=1 ttl=55 time=6.08 ms
64 bytes from 14.215.177.38: icmp_seq=2 ttl=55 time=6.21 ms
64 bytes from 14.215.177.38: icmp_seq=3 ttl=55 time=6.14 ms
64 bytes from 14.215.177.38: icmp_seq=4 ttl=55 time=5.52 ms
64 bytes from 14.215.177.38: icmp_seq=5 ttl=55 time=5.85 ms
^C
--- www.baidu.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4376ms
rtt min/avg/max/mdev = 5.527/5.966/6.214/0.259 ms
root@ATK-IMX6U:~#
```

图 3.11 6 测试 eth1 联网

查看网关后，并 ping 网关。

```
route
```

```
root@ATK-IMX6U:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth1
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth1
ipv4.connman.ne 192.168.1.1     255.255.255.255 UGH   0      0        0 eth1
root@ATK-IMX6U:~#
```

图 3.11 7 查看网关

由上可知网关为 192.168.1.1，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。

ping 192.168.1.1

```
root@ATK-IMX6U:~#
root@ATK-IMX6U:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=128 time=0.282 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=128 time=0.247 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=128 time=0.246 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=128 time=0.253 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=128 time=0.265 ms
^C
--- 192.168.1.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.246/0.258/0.282/0.022 ms
root@ATK-IMX6U:~#
```

图 3.11 8 ping 网关

网络通信速度测试

小提示：

iperf3 是一个网络性能测试工具。iperf 可以测试最大 TCP 和 UDP 带宽性能，具有多种参数和 UDP 特性，可以根据需要调整，可以报告带宽、延迟抖动和数据包丢失。

测试 Ubuntu 与开发板通信速度。如果你的 Ubuntu 未安装 iperf3，请在 Ubuntu 终端中执行"sudo apt-get install iperf3"安装。

查看 Ubuntu 的 ip 地址，备用。（这里要确保开发板的 ip 地址要与 Ubuntu 的 ip 地址是同

一局域网内）。

本次测试 Ubuntu 作服务端，开发板作客户端，执行下面指令。

```
ifconfig
```

```
alientek@ubuntu:~$ ifconfig
ens33     Link encap:以太网  硬件地址 00:0c:29:b0:56:0a
          inet 地址:192.168.1.166  广播:192.168.1.255  掩码:255.255.255.0
          inet6 地址: fe80::9bcc:8892:2471:2403/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1
          接收数据包:1750036 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:461086 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:985853212 (985.8 MB)  发送字节:882198465 (882.1 MB)

lo        Link encap:本地环回
          inet 地址:127.0.0.1  掩码:255.0.0.0
          inet6 地址: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1
          接收数据包:859 错误:0 丢弃:0 过载:0 帧数:0
          发送数据包:859 错误:0 丢弃:0 过载:0 载波:0
          碰撞:0 发送队列长度:1000
          接收字节:76616 (76.6 KB)  发送字节:76616 (76.6 KB)

alientek@ubuntu:~$
```

图 3.11 9 查看 Ubuntu 的 ip 地址

```
iperf3 -s        // Ubuntu 作为服务端
```

```
alientek@ubuntu:~$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

图 3.11 10 设置 Ubuntu 作为服务端

开发板作为客户端连接 Ubuntu 服务端。

```
iperf3 -c 192.168.1.84 -i 1          // -i 1 指通信周期，单位秒。
```

```
root@ATK-IMX6U:~# iperf3 -c 192.168.1.166 -i 1
Connecting to host 192.168.1.166, port 5201
[  4] local 192.168.1.244 port 45575 connected to 192.168.1.166 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  11.8 MBytes  98.9 Mbits/sec    0    232 KBytes
[  4]   1.00-2.00   sec  11.1 MBytes  93.3 Mbits/sec    0    245 KBytes
[  4]   2.00-3.00   sec  11.2 MBytes  93.8 Mbits/sec    0    245 KBytes
[  4]   3.00-4.00   sec  11.1 MBytes  93.3 Mbits/sec    0    245 KBytes
[  4]   4.00-5.00   sec  11.2 MBytes  94.4 Mbits/sec    0    245 KBytes
[  4]   5.00-6.00   sec  11.3 MBytes  94.9 Mbits/sec    0    245 KBytes
[  4]   6.00-7.00   sec  11.3 MBytes  94.8 Mbits/sec    0    253 KBytes
[  4]   7.00-8.00   sec  11.2 MBytes  93.9 Mbits/sec    0    257 KBytes
[  4]   8.00-9.00   sec  11.2 MBytes  93.8 Mbits/sec    0    257 KBytes
[  4]   9.00-10.00  sec  11.2 MBytes  93.8 Mbits/sec    0    257 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   113 MBytes  94.5 Mbits/sec    0             sender
[  4]   0.00-10.00  sec   112 MBytes  93.9 Mbits/sec                  receiver

iperf Done.
root@ATK-IMX6U:~#
```

图 3.11 11 开发板作为客户端并连接 Ubuntu 服务端

Ubuntu 服务端打印如下信息。

```
alientek@ubuntu:~$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.1.244, port 45574
[  5] local 192.168.1.166 port 5201 connected to 192.168.1.244 port 45575
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  10.7 MBytes  89.7 Mbits/sec
[  5]   1.00-2.00   sec  11.2 MBytes  93.8 Mbits/sec
[  5]   2.00-3.00   sec  11.2 MBytes  93.8 Mbits/sec
[  5]   3.00-4.00   sec  11.2 MBytes  93.8 Mbits/sec
[  5]   4.00-5.00   sec  11.2 MBytes  93.9 Mbits/sec
[  5]   5.00-6.00   sec  11.2 MBytes  93.8 Mbits/sec
[  5]   6.00-7.00   sec  11.2 MBytes  93.9 Mbits/sec
[  5]   7.00-8.00   sec  11.2 MBytes  93.7 Mbits/sec
[  5]   8.00-9.00   sec  11.2 MBytes  93.9 Mbits/sec
[  5]   9.00-10.00  sec  11.2 MBytes  93.9 Mbits/sec
[  5]  10.00-10.05  sec   580 KBytes  92.7 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.05  sec   113 MBytes  94.0 Mbits/sec    0             sender
[  5]   0.00-10.05  sec   112 MBytes  93.4 Mbits/sec                  receiver
```

图 3.11 12 Ubuntu 服务端打印的信息

反过来 Ubuntu 作客户端，开发板作服务端是一样的结果，这里就不浪费笔墨了。

3.12 FlexCAN 测试

开发板底板有一路 CAN（注：核心板支持两路，底板只能用一路，另一路被复用了）。要想测试 CAN，用户手上需要有测试 CAN 的仪器，（否则需要用两块不同的开发板的 CAN 或者其他 CAN 设备测试)比如周立功的 CAN 分析仪、创芯科技的 CAN 分析仪和广成科技的 CAN 分析仪等。或者可用两个 CAN 设备对接相互收发。关于 CAN 仪器及 CAN 上位机的使用，请参照各厂商的使用说明书，如不会使用请咨询 CAN 分析仪厂家的技术支持。

CAN 接口如下图，下图为 ALPHA 底板的位置，Mini 底板 CAN 接口位置的不一样。

3.12 1 ALPHA 底板的 CAN 接口

测试前请使用 CAN 分析仪或者测试 CAN 的设备连接好 I.MX6U 底板的 CAN，CANH 接仪器的 CANH，CANL 接 CAN 仪器的 CANL。

设置 can0 的 can 设备通信波特率为 1000000，也就是最大通信波特率 1MBit/s。

```
ip link set can0 up type can bitrate 1000000 triple-sampling on
```

```
root@ATK-IMX6U:~# ip link set can0 up type can bitrate 1000000 triple-sampling on
[  291.458511] flexcan 2090000.can can0: writing ctrl=0x01232084
[  291.465926] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
root@ATK-IMX6U:~#
```

3.12 2 打开 can 设备并设通信波特率

使用 candump 指令接收来自 can0 的数据

（1） -ta:t 代表打印时间，a 代表开启 ASCII 输出

```
candump -ta can0
```

使用 cansend 指令发送数据。

```
cansend can0 123#01.02.03.04.05.06.07.08
```

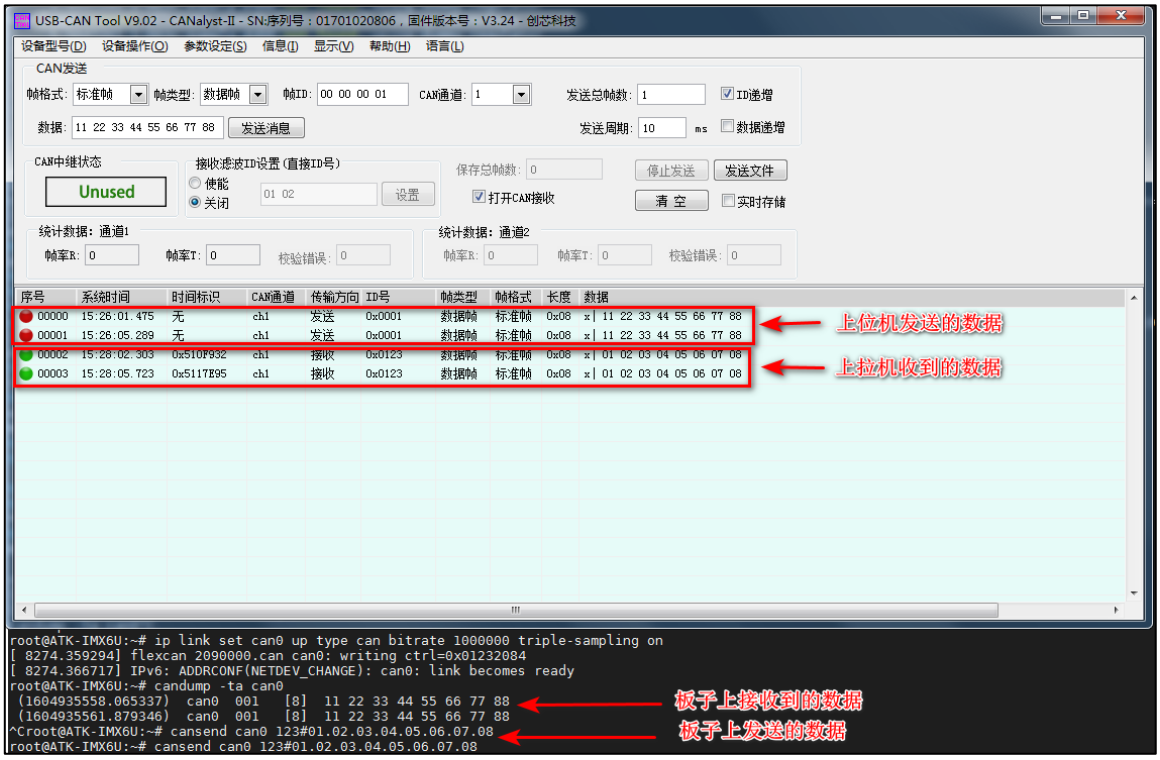
解释：

（1） can0 设备

（2） 123：帧 ID

（3） 01.02.03.04.05.06.07.08：帧数据

如下图，使用创芯科技的 CAN 分析仪上面机界面与开发板收发数据如下。



3.12 3 上位机与底板收发数据演示

3.13 USB 接口测试

ALPHA 底板 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

Mini 底板只有一个 USB_HOST

3.13.1 HOST 模式读写测试

以下实验根据使用的 U 盘或 TF 卡的不同和实验环境不同，测试结果会有所差异。

将 U 盘 或 TF 卡用读卡器插到开发板 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@ATK-IMX6U:~# [   43.715937] random: nonblocking pool is initialized
[   70.732844] usb 2-1.3: new high-speed USB device number 3 using ci_hdrc
[   70.847704] usb-storage 2-1.3:1.0: USB Mass Storage device detected
[   70.857369] scsi host0: usb-storage 2-1.3:1.0
[   71.863918] scsi 0:0:0:0: Direct-Access     MXT-USB  Storage Device   1501 PQ: 0 ANSI: 0 CCS
[   71.882023] sd 0:0:0:0: [sda] 30560256 512-byte logical blocks: (15.6 GB/14.5 GiB)
[   71.891003] sd 0:0:0:0: [sda] Write Protect is off
[   71.897246] sd 0:0:0:0: [sda] No Caching mode page found
[   71.902707] sd 0:0:0:0: [sda] Assuming drive cache: write through
[   71.925629] sd 0:0:0:0: [sda] Attached SCSI removable disk
[   72.246982] EXT4-fs (sda): recovery complete
[   72.251337] EXT4-fs (sda): mounted filesystem with ordered data mode. Opts: (null)

root@ATK-IMX6U:~#
root@ATK-IMX6U:~#
```

图 3.13.1 1 挂载的 U 盘信息

使用 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 sda。

```
df -h
```

```
root@ATK-IMX6U:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        15G  608M   14G   5% /
devtmpfs        184M  4.0K  184M   1% /dev
tmpfs            40K     0   40K   0% /mnt/.psplash
tmpfs           248M  172K  248M   1% /run
tmpfs           248M  140K  248M   1% /var/volatile
/dev/mmcblk1p2  7.0G  541M  6.1G   9% /run/media/mmcblk1p2
/dev/mmcblk0p1   63M   60M  3.7M  95% /run/media/mmcblk0p1
/dev/mmcblk1p1  127M  7.8M  119M   7% /run/media/mmcblk1p1
/dev/sda         15G   41M   14G   1% /run/media/sda
root@ATK-IMX6U:~#
```

图 3.13.1 2 查看 U 盘的挂载路径

写速度测试：

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

```
root@ATK-IMX6U:~# time dd if=/dev/zero of=/run/media/sda/test bs=1024k count=100 conv=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 6.60167 s, 15.9 MB/s

real    0m6.612s
user    0m0.000s
sys     0m1.330s
root@ATK-IMX6U:~#
```

图 3.13.1 3 写速度测试

本次写 100MiB，速度为 15.9 MB/s。

读速度测试：

小提示：

因为 LINUX 的内核机制，一般情况下不需要特意去释放已经使用的 cache。这些 cache 内容可以增加文件的读写速度。

执行下面指令清除缓存。

```
echo 3 > /proc/sys/vm/drop_caches
```

执行下面的指令读取前面用 dd 指令写入的 test 文件。

```
time dd if=/run/media/sda1/test of=/dev/null bs=1024k
```

```
root@ATK-IMX6U:~# echo 3 > /proc/sys/vm/drop_caches
[  434.961503] sh (684): drop_caches: 3
root@ATK-IMX6U:~# time dd if=/run/media/sda/test of=/dev/null bs=1024k
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 5.77877 s, 18.1 MB/s

real    0m5.814s
user    0m0.000s
sys     0m0.540s
root@ATK-IMX6U:~#
```

图 3.13.1 4 读速度测试

这里一共读出了 100 MiB test 文件，速度为 18.1 MB/s。

3.13.2 DEVICE 模式测试

本实验将 TF 卡的第一个分区“boot”分区模拟成 U 盘，请使用一根 USB 转串口线接在 USB_OTG1 处，并连接 PC 端 USB 接口。在串口终端执行下面指令，就可以将开发板的 TF 卡模拟成 U 盘挂载在 PC 上。

```
modprobe g_mass_storage file=/dev/mmcblk0p1 removable=1
```

```
root@ATK-IMX6U:~# modprobe g_mass_storage file=/dev/mmcblk0p1 removable=1
[   21.181700] Mass Storage Function, version: 2009/09/11
[   21.187133] LUN: removable file: (no medium)
[   21.191644] LUN: removable file: /dev/mmcblk0p1
[   21.197256] Number of LUNs=1
[   21.200175] Number of LUNs=1
[   21.204357] g_mass_storage gadget: Mass Storage Gadget, version: 2009/09/11
[   21.211350] g_mass_storage gadget: userspace failed to provide iSerialNumber
[   21.219357] g_mass_storage gadget: g_mass_storage ready
root@ATK-IMX6U:~# [   43.953570] g_mass_storage gadget: high-speed config #1: Linux File-Backed Storage
```

图 3.13.2 1 将开发板 boot 分区模拟成一个 U 盘

如下图，已经将 TF 卡的第一个分区成功的挂载在 PC 上。可以当作 U 盘一样使用。

图 3.13.2 2 模拟成一个 U 盘成功

3.13.3 USB SERIAL 测试

本实验将 USB_OTG1 当作串口使用。将另一根 USB 转串口线接到 USB_OTG1 处，然后连接 PC，在串口终端下执行如下指令。(注:如果使用了上小节的 DEVICE 模式，需要重启开发板再做此实验)。

```
modprobe g_serial
```

```
root@ATK-IMX6U:~# modprobe g_serial
[   78.808506] g_serial gadget: Gadget Serial v2.4
[   78.816514] g_serial gadget: g_serial ready
root@ATK-IMX6U:~#
```

图 3.13.3 1 将 USB_OTG1 模拟成一个串口

查看是否生成/dev/ttyGS0 节点。

```
ls /dev/ttyGS0
```

```
root@ATK-IMX6U:~# ls /dev/ttyGS0
/dev/ttyGS0
root@ATK-IMX6U:~#
```

图 3.13.3 2 查看生成的节点

同时可以在 PC 设备管理器处，查看端口号

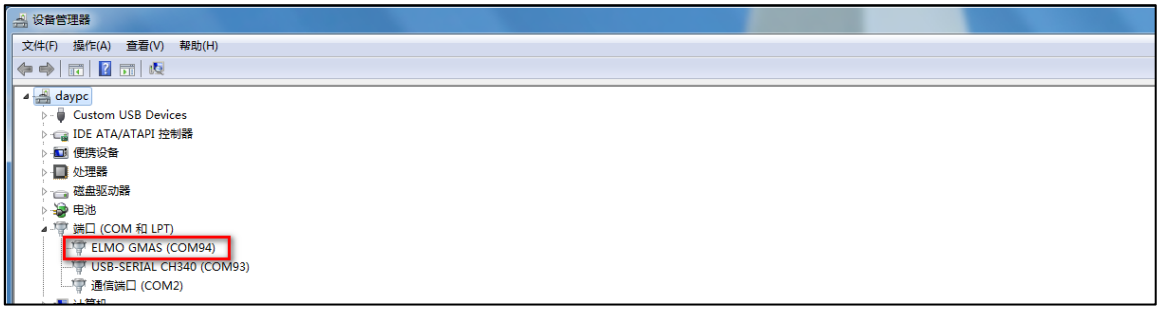


图 3.13.3. 3 在设备管理器查看端口号

开启守护进程。

```
setsid getty 115200 /dev/ttyGS0
```

```
root@ATK-IMX6U:~# setsid getty 115200 /dev/ttyGS0
root@ATK-IMX6U:~#
```

图 3.13.3 4 开启守护进程，设置模拟成一个串口终端

然后用 SecureCRT 或者 MobaXterm 软件工具连接到该端口号。打开串口调试终端，选择正确的 COM 口，波特率为 115200，8N1，无检验位，并建立串口连接，按下回车键，可以后其他串口终端一样使用了。

3.14 USB 鼠标测试

说明：使用出厂文件系统，启动前插上 RGB LCD 屏幕。

开发板进入系统后，插上鼠标会打印如下信息。

```
root@ATK-IMX6U:~# [  758.052821] usb 1-1.4: new low-speed USB device number 4 using ci_hdrc
[  758.175676] input:  USB OPTICAL MOUSE as /devices/platform/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb1/1-1/1-1.4/1-1.4:1.0/0003:0000:3825.0001/input/input2
[  758.192011] hid-generic 0003:0000:3825.0001: input: USB HID v1.11 Mouse [ USB OPTICAL MOUSE] on usb-ci_hdrc.1-1.4/input0

root@ATK-IMX6U:~#
```

图 3.14 1 插上鼠标后打印的信息

同时，出厂 Qt 桌面也显示了鼠标，可以使用鼠标进行点击操作。

3.15 音频测试

ALPHA	MINI
支持本实验	本实验不支持，底板无音频，但是可接 usb 声卡

3.15.1 ALSA 简单使用

ALSA（高级 Linux 声音架构）在 Linux 操作系统上提供了音频和 MIDI（Musical Instrument Digital Interface，音乐设备数字化接口）的支持。

amixer 的使用：

```
amixer --help          // 查看 amixer 的用法说明
```

```
root@ATK-IMX6U:~# amixer --help
Usage: amixer <options> [command]

Available options:
  -h,--help       this help
  -c,--card N     select the card
  -D,--device N   select the device, default 'default'
  -d,--debug      debug mode
  -n,--nocheck    do not perform range checking
  -v,--version    print version of this program
  -q,--quiet      be quiet
  -i,--inactive   show also inactive controls
  -a,--abstract L select abstraction level (none or basic)
  -s,--stdin      Read and execute commands from stdin sequentially
  -R,--raw-volume Use the raw value (default)
  -M,--mapped-volume Use the mapped volume

Available commands:
  scontrols       show all mixer simple controls
  scontents       show contents of all mixer simple controls (default command)
  sset sID P      set contents for one mixer simple control
  sget sID        get contents for one mixer simple control
  controls        show all controls for given card
  contents        show contents of all controls for given card
  cset cID P      set control contents for one control
  cget cID        get control contents for one control
root@ATK-IMX6U:~#
```

图 3.15.1 1 查看 amixer 的帮助信息

alsamixer 的使用：

这里不得不说 ALSA 真的是强大啊，amixer 是用指令控制音频设备，alsamixer 则提供一套图形界面来控制音频设备，可以用键盘方向键来控制增减音量，打开或者关闭等。

输入下面的指令，打开 alsamixer 图形界面，可用键盘的左右上下方向键等键操作，退出直接按 Esc 按键。下图柱形图就是 wm8960-audio 的音频控制界面。

```
alsamixer
```

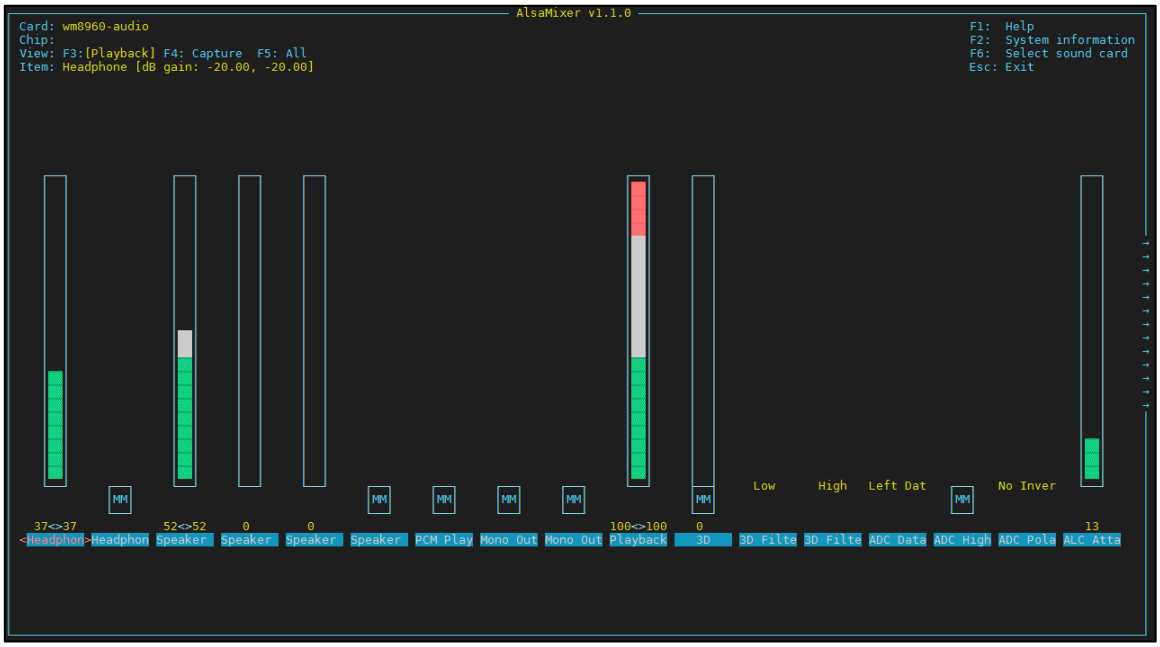


图 3.15.1 2 alsamixer 图形控制界面

3.15.2 Headphone 测试

开发板系统音频输出功能默认是打开的，下面两条指令可不执行。

```
amixer sset 'Left Output Mixer PCM' on
amixer sset 'Right Output Mixer PCM' on
```

播放音频文件时插耳机到 PHONE 接口处，可以通过下面的指令来设置耳机的音量，或者通过 alsamixer 图形界面的方法来设置音量。

设置播放音量，执行如下命令，音量的单位是 dB，音量最小为 0，最大为 127。

```
amixer sset Headphone 110,110                    // 耳机音量设置为 52
```

播放音频

播放开发板文件系统自带的音频，执行下面指令

```
aplay /usr/share/sounds/alsa/Front_Center.wav
aplay /usr/share/sounds/alsa/Front_Left.wav
aplay /usr/share/sounds/alsa/Front_Right.wav
```

```
root@ATK-IMX6U:~# aplay /usr/share/sounds/alsa/Front_Center.wav
Playing WAVE '/usr/share/sounds/alsa/Front_Center.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Mono
root@ATK-IMX6U:~# aplay /usr/share/sounds/alsa/Front_Left.wav
Playing WAVE '/usr/share/sounds/alsa/Front_Left.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Mono
root@ATK-IMX6U:~# aplay /usr/share/sounds/alsa/Front_Right.wav
Playing WAVE '/usr/share/sounds/alsa/Front_Right.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Mono
root@ATK-IMX6U:~#
```

图 3.15.2 1 使用 aplay 播放音频文件

3.15.3 Speaker 测试

ALPHA 开发板底板留出 2 路扬声器接口，一路是接左声道（SPKL），一路是接右声道（SPKR），其中底板的扬声器（小喇叭 8 欧 2W）已经接在右声道（SPKR）上。

ALPHA 开发板底板使用的音频芯片是 WM8960 ，是一款低功耗立体声编解码器，采用 D 类扬声器驱动器，可在 8 欧负载下为每通道提供大于 1 W 功率。

注：测试外接扬声器（喇叭时），不要接插入耳机。

可不执行下面的指令设置扬声器的音量，默认音量为 86。

```
amixer sset Speaker 110,110                    // 扬声器（喇叭）的音量设置为 52
```

同上面 Headphone 测试一样，执行播放音频的指令测试有声音出来即可。

3.15.4 LINE IN 音频输入测试

为了方便，正点原子提供设置音频输入的脚本和录音脚本，脚本里有注释，脚本内容仅供用户参考。

LINE IN 录音内部无信号放大电路，使用一条 3.5mm 两头均为公头的音频线，一端连接开发板的 LINE IN 接口，另一端连接正在播放音乐的 PC 机或手机。

```
cd shell/audio/
./line_in_config.sh
```

```
root@ATK-IMX6U:~# cd shell/audio/
root@ATK-IMX6U:~/shell/audio# ./line_in_config.sh
numid=1,iface=MIXER,name='Capture Volume'
  ; type=INTEGER,access=rw---R--,values=2,min=0,max=63,step=0
  : values=63,63
  | dBscale-min=-17.25dB,step=0.75dB,mute=0
amixer: Unable to find simple control 'PCM Playback',0

Simple mixer control 'Playback',0
  Capabilities: volume
  Playback channels: Front Left - Front Right
  Capture channels: Front Left - Front Right
  Limits: 0 - 255
  Front Left: 255 [100%] [0.00dB]
  Front Right: 255 [100%] [0.00dB]
Simple mixer control 'Headphone Playback ZC',0
```

图 3.15.4 1 执行 line_in_config.sh 配置音频输入

执行 record.sh 开始录音（注 record.sh 会录一段 10 秒钟的录音保存为当前目录下名为 record.wav 的音频文件，录制音频完成后会自动播放 record.wav，如果音频播放很小声，请将输入端的音量调高）。

```
./record.sh
```

图 3.15.4 2 执行 record.sh 开始录音，录音完成自动播放

3.15.5 MIC IN 录音测试

为了方便，正点原子提供测试麦克风输入的脚本和录音脚本，脚本里有注释，脚本内容仅供用户参考。

```
./mic_in_config.sh
```

```
root@ATK-IMX6U:~/shell/audio# ./mic_in_config.sh
numid=1,iface=MIXER,name='Capture Volume'
  ; type=INTEGER,access=rw---R--,values=2,min=0,max=63,step=0
  : values=63,63
  | dBscale-min=-17.25dB,step=0.75dB,mute=0
amixer: Unable to find simple control 'PCM Playback',0

Simple mixer control 'Playback',0
  Capabilities: volume
  Playback channels: Front Left - Front Right
  Capture channels: Front Left - Front Right
  Limits: 0 - 255
  Front Left: 255 [100%] [0.00dB]
  Front Right: 255 [100%] [0.00dB]
Simple mixer control 'Headphone Playback ZC',0
```

图 3.15.5 1 执行 mic_in_config.sh 配置麦克风输入

执行 record.sh 开始录音（注 record.sh 会录一段 10 秒钟的录音保存为当前目录下名为 record.wav 的音频文件，录制音频完成后会自动播放 record.wav，请对准备开发板底板上的麦头进行大声说话进行录音）。

```
./record.sh
```

图 3.15.5 2 执行 record.sh 开始录音，录音完成自动播放

3.16 ov5640/ov2640/ov7725 摄像头测试

正点原子 I.MX6ULL 开发板支持正点原子店铺的三款摄像头模块。ov5640（500 万像素），ov2640(200 万像素)，ov7725（不带 FIFO 款）(30 万像素)。可以在正点原子店铺选购。支持 YUYV 及 RGB565 等格式采集。

3.16.1 ov5640 摄像头

实验前请准备 ov5640 摄像头模块（500 万像素），本公司的任何分辨率的 RCB LCD 电容屏。摄像头美照如下。

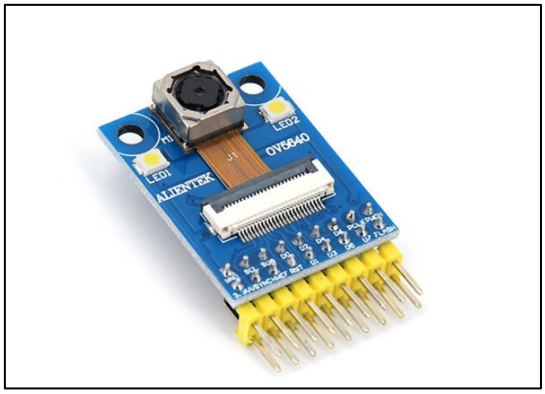


图 3.16.1 1 正点原子 ov5640 模块

摄像头插法：

摄像头镜头往开发板外则直接插到 CAMERA 接口处。由于没有防反插设计，插摄像头时需要注意看底板丝印，按引脚编号对应插上。插上效果如下(下图为 ALPHA 开发板接法，Mini Linux 开发板也一样，镜头朝外侧即可)。**注意不要与摄像头延长线连接，否则因延长线太长，导致信号干扰。**

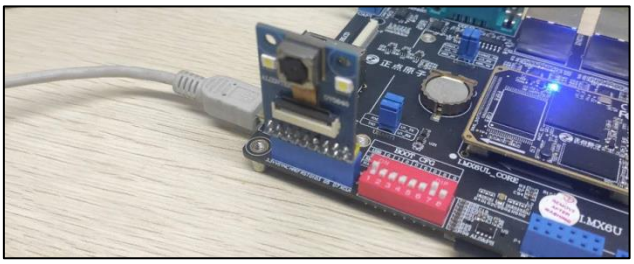


图 3.16.1 2 ov5640 与 AlPHA 开发板接法

本实验使用本公司的 ov5640 模块通过 CSI 总线对视频进行实时采集。

插拔 ov5640 摄像头需要注意：

◆ 该摄像头不支持热插拔，所以在插拔时都需要断电，再进行操作。

◆ 插摄像头时请注意底板上的丝印要与摄像头上面的丝印对上，再插上摄像头。

ov5640 在内核里是编译成模块形式的，插上 ov5640 摄像头，板子启动时可以看到如下信息。

```
bootlogd: cannot allocate pseudo tty: No such file or directory
[    6.035267] 1-003c supply DOVDD not found, using dummy regulator
[    6.041461] 1-003c supply DVDD not found, using dummy regulator
[    6.172261] 1-003c supply AVDD not found, using dummy regulator
[    7.622773] CSI: Registered sensor subdevice: ov5640 1-003c
[    7.628370] camera ov5640, is found
```

图 3.16.1 3 ov5640 驱动模块加载信息

查看是否生成 video1 节点。（这里可能是 video2，用户根据实际情况查看/dev 下的 video 设备）。

注：一般是 video1 节点就是 ov5640 摄像头的节点，但是也会有特殊情况（需要看 video 设备驱动的加载顺序）。

```
ls /dev/video1
```

```
root@ATK-IMX6U:~# ls /dev/video1
/dev/video1
root@ATK-IMX6U:~#
```

图 3.16.1 4 查看 ov5640 的节点

查看出厂内核版本，在正点原子出厂内核 v1.6 版本之前（请在 1.2.2 小节查看出厂内核历史版本，检查用户所处的固件版本）是不支持 RGB565 和 JPEG 采集的。同时在 v1.6 版本的内核添加以正点原子屏为大小的摄像头采集分辨率，如下图图 3.16.1 4。其中 480x272，800x480，1024x600 和 1280x800 都是正点原子 RGB 屏的分辨率，设置这样的一个非标准采集分辨率是为了方便写应用程序，同时在有限的屏幕大小内，以相同的分辨率采集可节省显示性能。

查看摄像头支持格式、分辨率及帧率。

```
v4l2-ctl --device=/dev/video1 --list-formats-ext
```

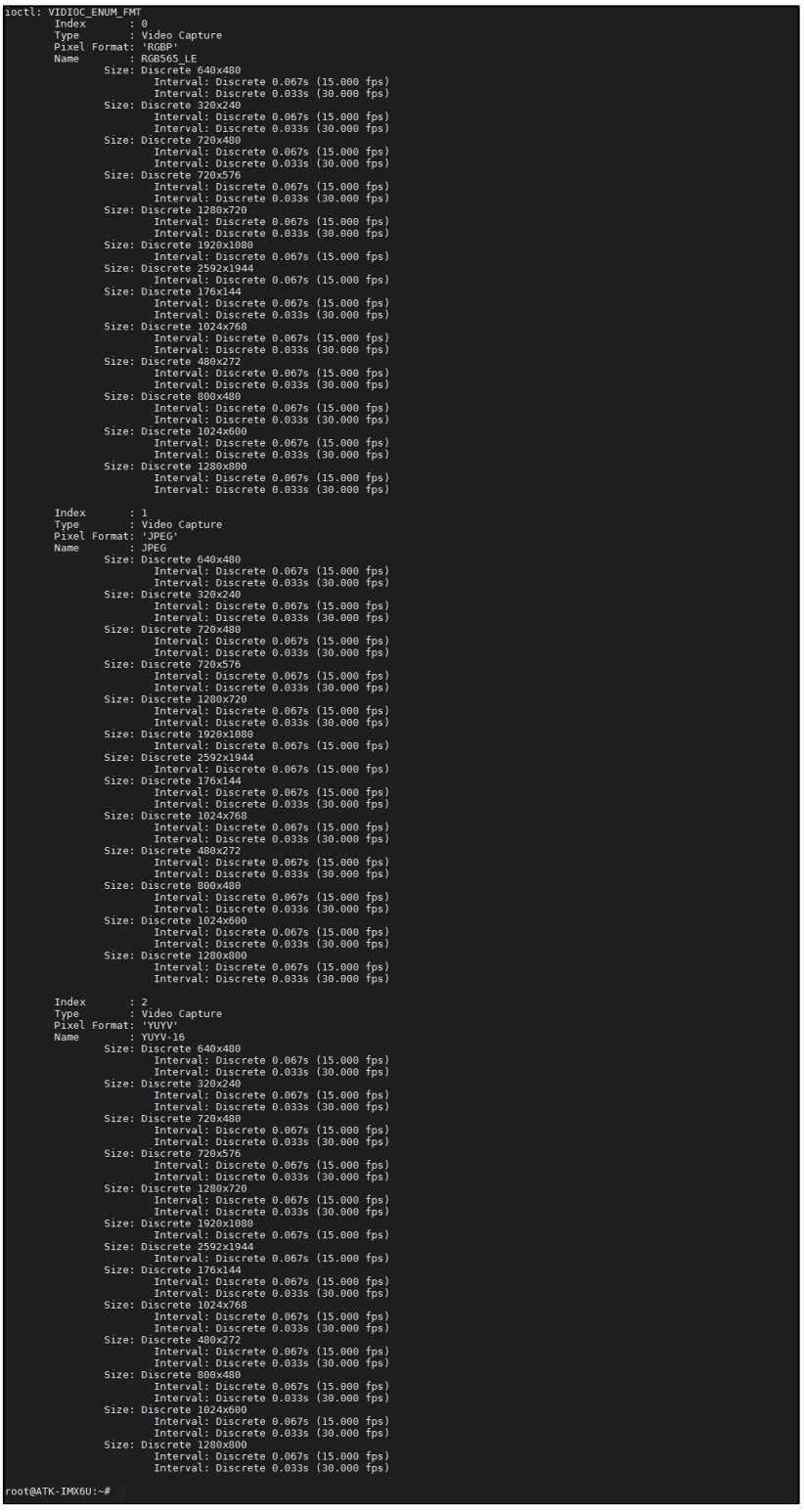


图 3.16.1 4 查看摄像头所支持的格式

3.16.1.1 使用 gstreamer 采集

- 采集图像并显示

本次设置采集的图像分辨率为 1024*768 30fps，如果需要设置其他分辨率采集需要严格按照上面的参数。执行下面指令开始采集，并显示到 LCD 上面，按“Ctrl + c”快捷键终止指令，停止采集。由于使用 gstreamer 使用元件 imxv4l2src 不能设置 JPEG 格式采集，它只能设置 YUYV 格式和 RGB565 模式采集。YUYV 格式传入字符参数是 format=(string)YUY2，RGB565 格式，传入的参数为 **RGB16**。（注：下面为一行指令，由于 pdf 格式问题，复制时建议分段复制）YUYV 格式采集（测试前请退出 Qt 桌面。）：

```
gst-launch-1.0 -v imxv4l2src device=/dev/video1 ! "video/x-raw, format=(string)YUY2, width=(int)1024, height=(int)768, framerate=(fraction)30/1" ! imxv4l2sink
```

RGB565 格式采集（测试前请退出 Qt 桌面。）：

```
gst-launch-1.0 -v imxv4l2src device=/dev/video1 ! "video/x-raw, format=(string)RGB16, width=(int)1024, height=(int)768, framerate=(fraction)30/1" ! imxv4l2sink
```

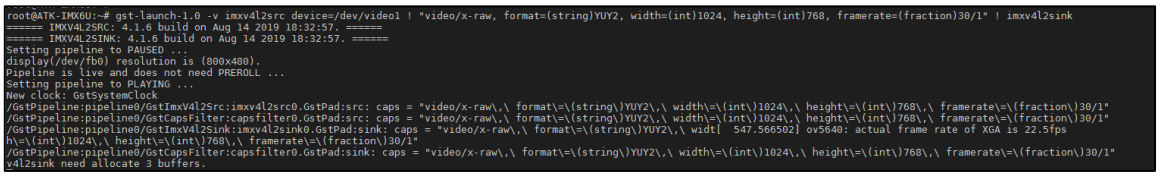


图 3.16.1.1 1 执行指令采集图像

屏幕使用本公司 7 寸 800*480 RGB 屏，采集图像十分流畅。图像如下：

图 3.16.1.1 2 7 寸屏上的图像

- 保存视频

使用 ov5640 保存视频，本次就不作演示了。指令参考如下，注意对应 ov5640 的节点。录像完成后会在当前目录下保存一个 video.yuv 的视频文件。把它上传到电脑使用 yuv 相关的播放器播放即可。由于 pdf 格式问题，复制时建议段复制）。

保存为 yuv 格式的视频

```
gst-launch-1.0 -vvv -e imxv4l2src num-buffers=1000 device=/dev/video1 ! "video/x-raw, format=(string)YUY2, width=(int)1024, height=(int)768, framerate=(fraction)30/1" ! filesink location=video.yuv
```

保存为 avi 格式的视频，由于用到 jpegenc 转换编码，性能不足，保存的慢，但是播放的快。所以保存的视频会有加快的现象。

```
gst-launch-1.0 -vvv -e imxv4l2src num-buffers=1000 device=/dev/video1 ! "video/x-raw, format=(string)YUY2, width=(int)320, height=(int)240, framerate=(fraction)30/1" ! jpegenc ! filesink location=video.avi
```

- 拍照功能

使用下面的指令拍照，保存图像为 yuv 格式（注：下面为一行指令，由于 pdf 格式问题，复制时建议分段复制）

```
gst-launch-1.0 imxv4l2src num-buffers=1 device=/dev/video1 ! 'video/x-raw,format=(string)YUY2,width=1024,height=768' ! filesink location=picture.yuv
```

使用下面的指令拍照，保存图像为 jpg 格式（注：下面为一行指令，由于 pdf 格式问题，复制时建议分段复制）

```
gst-launch-1.0 imxv4l2src num-buffers=1 device=/dev/video1 ! jpegenc ! filesink location=picture.jpg
```

3.16.1.2 使用 ffmpeg 采集

由于 ffmpeg 不能直接设置 ov5640 摄像头的采集分辨率。所以正点原子编写了一个 camera_settings 应用指令，放在/usr/bin/目录下。关于这个应用指令的使用方法如下。使用 camera_settings 应用指令可以设置 ov5640 的采集分辨率和采集格式，也可以设备 USB 摄像头的采集分辨率和采集格式。

```
camera_settings -help
```

```
root@ATK-IMX6U:~# camera_settings -help

Usage: camera_setting [device] [format] [width] [height] [frame]

video device
/dev/videoX

video format
RGB565 | YUYV | JPEG | MJPG

video frame
15 | 30 | 10 | other

example
camera_settings /dev/video1 RGB565 320 240 30

root@ATK-IMX6U:~#
```

图 3.16.1.2 1 camera_settings 的使用帮助

例：

```
camera_settings /dev/video1 RGB565 320 240 30
```

指令解释：

（1） camera_settings: 正点原子编写的应用指令。
（2） /dev/video1: ov5640 设备节点
（3） RGB565: 设置摄像头的采集格式，ov5640 支持 RGB565，YUYV，JPEG 三种格式
（4） 320: 设置采集的宽度
（5） 240: 设置采集的高度
（6） 30: 帧率，支持 15 帧和 30 帧

按上面的指令设置完成后 ov5640 摄像头的采集分辨率后，采集的图像分辨率越大，可能采集时性能不足，所以我们设置一个比较低的分辨率采集。我们就可以使用 ffmpeg 指令来保存视频等操作了。

注意，如果用户使用的是 NandFlash 类型的核心板（512MB 存储大小，文件系统功能越来越丰富，NandFlash 存储容量可能会不够），录制视频会占用大量存储量空间，请插入 TF 卡或者 U 盘都到开发板上。使用 cd 命令进入到 TF 卡或者 U 盘的挂载目录下，一般是/run/media/sda1。这样视频就会保存到 TF 卡或者 U 盘的目录。

➢ 保存视频，保存在当前目录。

```
ffmpeg -t 10 -pix_fmt rgb565le -i /dev/video1 video.avi
```

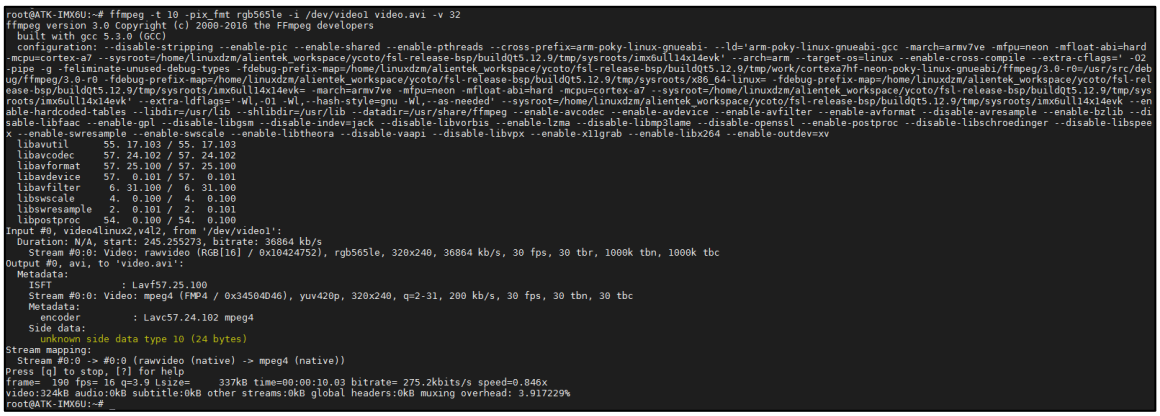


图 3.16.1.2 2 使用 ffmpeg 保存 ov5640 摄像头录像视频

指令解释：

（1） ffmpeg: ffmpeg 指令

（2） -t 10: 设置采集时间为 10s

（3） -pix_fmt rgb565le: 设置采集的格式，（RGB565）rgb565le 采集，此外支持(YUYV)yuyv422 和（JPEG）yuyv422 采集（这里为什么 YUYV 和 JPEG 一样是 yuyv422，编者研究源码得出，暂时未确保准确性）。

（4） -i /dev/video1: 指定 ov5640 设备节点。

（5） video.avi: 输出文件名，不同的格式会自动选择不同的编码，目前测试 avi 格式流畅。

播放采集的录像

```
gst-play-1.0 video.avi
```

使用 ffmpeg 设置编码指令请执行 ffmpeg -help 查看相关帮助与用法，本手册不再详细说明。

3.16.1.3 使用 ov5640_camera 采集

使用 ov5640_camera 指令（请在 1.2.2 小节查看出厂内核历史版本，检查用户所处的固件版本，ov5640_camera 在 v1.9 出厂文件系统版本才有）直接采集显示在正点原子 RGB LCD 屏上。ov5640_camera 是正点原子编写的摄像头应用程序，放在/usr/bin/目录下。这个 ov5640_camera 根据正点原子屏幕的分辨率大小，设置相应分辨率采集，并全屏显示在正点原子 RGB LCD 屏幕上，仅供测试使用。

在开发板直接执行 ov5640_camera，测试前请退出 Qt 桌面。采集的分辨率为屏幕的显示分辨率大小，帧数为 30 帧采集，采集显示在 LCD 屏上的图像非常流畅。

```
ov5640_camera
```

图 3.16.1.3 1 执行 ov5640_camera 应用程序

3.16.2 ov2640 摄像头

实验前请准备 ov2640 摄像头（200 万像素），可在正点原子店铺购买。ov2640 摄像头美照如下。

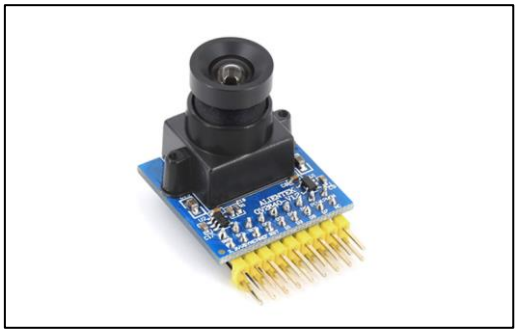


图 3.16.2 1 正点原子 ov2640 摄像头模块

由于出厂系统默认是配置了 ov5640 摄像头，所以我们要在出厂系统源码里修改设备树，关闭 ov5640 摄像头，开启 ov2640 摄像头才能进行此实验。修改内核源码的设备树，路径为内核源码顶层目录下的 arch/arm/boot/dts/imx6ull-14x14-evk.dts ，编辑**第 324 行及 347 行**，将“ON”改为 OFF 即将其关闭，将“OFF”改为“ON”即将其打开。修改完成如下。

```
324 #if ATK_CAMERA_OFF
325         ov5640: ov5640@3c {
326                 compatible = "ovti,ov5640";
327                 reg = <0x3c>;
328                 pinctrl-names = "default";
329                 pinctrl-0 = <&pinctrl_csi1
330                              &csi_pwn_rst>;
331                 clocks = <&clks IMX6UL_CLK_CSI>;
332                 clock-names = "csi_mclk";
333                 pwn-gpios = <&gpio1 4 1>;
334                 rst-gpios = <&gpio1 2 0>;
335                 csi_id = <0>;
336                 mclk = <24000000>;
337                 mclk_source = <0>;
338                 status = "okay";
339                 port {
340                         camera_ep: endpoint {
341                                 remote-endpoint = <&csi_ep>;
342                         };
343                 };
344         };
345 #endif
346
347 #if ATK_CAMERA_ON
348         ov2640: camera@0x30 {
349                 compatible = "ovti,ov2640";
350                 reg = <0x30>;
351                 status = "okay";
352
353                 pinctrl-names = "default";
```

```
354                 pinctrl-0 = <&pinctrl_csi1
355                             &csi_pwn_rst>;
356                 resetb = <&gpio1 2 GPIO_ACTIVE_LOW>;
357                 pwdn = <&gpio1 4 GPIO_ACTIVE_HIGH>;
358                 clocks = <&clks IMX6UL_CLK_CSI>;
359                 clock-names = "xvclk";
360
361                 port {
362                         camera_ep: endpoint {
363                                 remote-endpoint = <&csi_ep>;
364                                 bus-width = <8>;
365                         };
366                 };
367         };
368 #endif
```

修改完成后编译内核源码的详细请参考 4.4 小节。因为我们只修改了设备树，所以我们执行 build.sh 脚本编译后在 tmp 目录下找到所有的 dtb 文件，将其拷贝到开发板光盘 A-基础资料->5、开发工具->4、正点原子 MFG_TOOL 出厂固件烧录工具->mfgtool->Profiles->Linux->OS Firmware->files->boot 下的所有 dtb 文件。然后再按 2.2 小节重新固化系统即可！如果不想重新烧写系统，亦可按照【正点原子】I.MX6U 开发板文件拷贝及固件更新参考手册 V1.x.pdf 单独烧录相应的设备树即可。

重新烧写系统后，再插上 ov2640 摄像头，接法与 3.16.1 小节 ov5640 的接法一样。在打印信息处，驱动加载后如下。

```
[    3.301265] devtmpfs: mounted
[    3.305000] Freeing unused kernel memory: 528K (80b48000 - 80bcc000)
INIT: version 2.88 booting
Starting udev
[    3.918823] udevd[129]: starting version 3.1.5
[    3.933078] random: udevd urandom read with 26 bits of entropy available
bootlogd: cannot allocate pseudo tty: No such file or directory
[    4.894544] ov2640 1-0030: ov2640 Product ID 26:42 Manufacturer ID 7f:a2
[    4.965371] CSI: Registered sensor subdevice: ov2640 1-0030
[    4.970983] i2c i2c-1: OV2640 Probed
```

图 3.16.2 2 摄像头驱动加载打印的信息

查看摄像头支持格式。

```
v4l2-ctl --device=/dev/video1 --list-formats-ext
```

```
root@ATK-IMX6U:~# v4l2-ctl --device=/dev/video1 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUYV-16

        Index       : 1
        Type        : Video Capture
        Pixel Format: 'UYVY'
        Name        : UYVY-16

        Index       : 2
        Type        : Video Capture
        Pixel Format: 'RGBP'
        Name        : RGB565_BE

        Index       : 3
        Type        : Video Capture
        Pixel Format: 'RGBP'
        Name        : RGB565_LE

root@ATK-IMX6U:~#
```

图 3.16.2 3 查看驱动中正点原子 ov2640 摄像头支持的格式

3.16.2.1 使用 gstreamer 采集

➢ 采集图像并显示

可以设置小于最大采集分辨率的采集分辨率(1600x1200)。由于 pdf 格式问题，复制时分段复制，这里 **format=(string)RGB16** 即 RGB565 采集，可支持 **format=(string)YUY2** 即 YUYV 格式采集。

```
gst-launch-1.0 -v v4l2src device=/dev/video1 ! "video/x-raw, format=(string)RGB16, width=(int)352, height=(int)288" ! imxv4l2sink
```

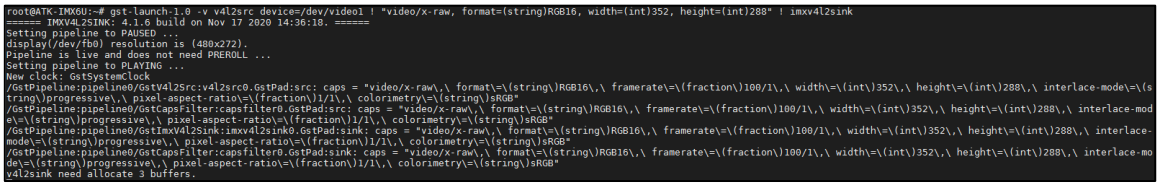


图 3.16.2.1 1 使用 gstreamer 采集

效果图略，十分流畅。

拍照功能等请参考上一节 ov5640 的相关指令。

3.16.2.2 使用 ov2640_camera 采集

ov2640_camera 应用程序是正点原子编写的一个用于测试的程序。在串口终端直接执行 ov2640_camera 即可。

ov2640_camera 如果是 4.3 寸屏 480x272 的屏幕可执行 ov772x_camera 指令。

图 3.16.2.2 1 使用正点原子编写的 ov2640_camera 采集

效果图略，十分流畅。

3.16.3 ov7725 摄像头

实验前准备 ov7725 摄像头（30 万像素）模块，30 万像素这个完全足够 I.MX6U 使用。最大采集分辨率为 640x480。在 I.MX6U 上缺点就是可支持的分辨率少。不如 ov5640 与 ov2640 摄像头模块在**驱动中**所支持的分辨率多，仅支持正点原子的 ov7725 模块**不带 FIFO 款**。摄像头模块美照如下。

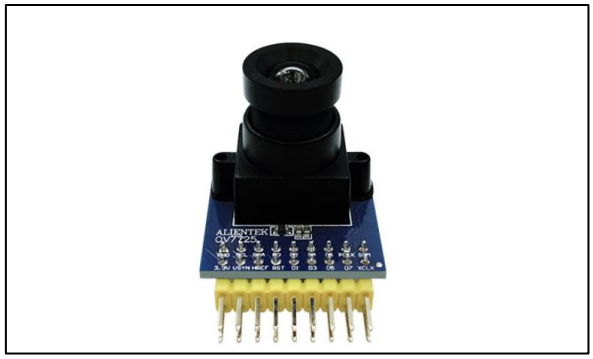


图 3.16.3 1 正点原子 ov7725 摄像头模块

由于出厂系统默认是配置了 ov5640 摄像头，所以我们要在出厂系统源码里修改设备树，关闭 ov5640 摄像头，开启 ov7725 摄像头才能进行此实验。修改内核源码的设备树，路径为内核源码顶层目录下的 arch/arm/boot/dts/imx6ull-14x14-evk.dts ，编辑**第 324 行及 370 行**，将“ON”改为 OFF 即将其关闭，将“OFF”改为“ON”即将其打开。修改完成如下。

```
324 #if ATK_CAMERA_OFF
325         ov5640: ov5640@3c {
326                 compatible = "ovti,ov5640";
327                 reg = <0x3c>;
328                 pinctrl-names = "default";
329                 pinctrl-0 = <&pinctrl_csi1
330                              &csi_pwn_rst>;
331                 clocks = <&clks IMX6UL_CLK_CSI>;
332                 clock-names = "csi_mclk";
333                 pwn-gpios = <&gpio1 4 1>;
334                 rst-gpios = <&gpio1 2 0>;
335                 csi_id = <0>;
336                 mclk = <24000000>;
337                 mclk_source = <0>;
338                 status = "okay";
339                 port {
340                         camera_ep: endpoint {
341                                 remote-endpoint = <&csi_ep>;
342                         };
343                 };
344         };
345 #endif
... ... 省略其他行
370 #if ATK_CAMERA_ON
371         ov7725: camera@0x21 {
372                 compatible = "ovti,ov772x","ovti,ov7725";
373                 reg = <0x21>;
374                 status = "okay";
375                 pinctrl-names = "default";
```

```
376                 pinctrl-0 = <&pinctrl_csi1
377                             &csi_pwn_rst>;
378                 resetb = <&gpio1 2 GPIO_ACTIVE_LOW>;
379                 pwdn = <&gpio1 4 GPIO_ACTIVE_HIGH>;
380                 clocks = <&clks IMX6UL_CLK_CSI>;
381                 clock-frequency = <20000000>;
382                 clock-names = "mclk";
383
384                 port {
385                         camera_ep: endpoint {
386                                 remote-endpoint = <&csi>;
387                                 bus-width = <8>;
388                         };
389                 };
390         };
391 #endif
```

修改完成后编译内核源码的详细请参考 4.4 小节。因为我们只修改了设备树，所以我们执行 build.sh 脚本编译后在 tmp 目录下找到所有的 dtb 文件，将其拷贝到开发板光盘 A-基础资料->5、开发工具->4、正点原子 MFG_TOOL 出厂固件烧录工具->mfgtool->Profiles->Linux->OS Firmware->files->boot 下的所有 dtb 文件。然后再按 2.2 小节重新固化系统即可！如果不想重新烧写系统，亦可按照【正点原子】I.MX6U 开发板文件拷贝及固件更新参考手册 V1.x.pdf 单独烧录相应的设备树即可。

重新烧写系统后，再插上 ov7725 摄像头，接法与 3.16.1 小节 ov5640 的接法一样。在打印信息处，驱动加载后如下。

```
[    3.347838] Freeing unused kernel memory: 528K (80b48000 - 80bcc000)
INIT: version 2.88 booting
Starting udev
[    3.998151] udevd[129]: starting version 3.1.5
[    4.012360] random: udevd urandom read with 28 bits of entropy available
bootlogd: cannot allocate pseudo tty: No such file or directory
[    4.932602] ov772x 1-0021: ov7725 Product ID 77:21 Manufacturer ID 7f:a2
[    5.008624] CSI: Registered sensor subdevice: ov772x 1-0021
[    5.055182] i2c i2c-1: OV772x Probed
[    7.631568] FAT-fs (mmcblk1p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
ALSA: Restoring mixer settings...
```

图 3.16.3 2 正点原子 ov7725 摄像头模块驱动加载的信息

查看摄像头支持的格式。

```
v4l2-ctl --device=/dev/video1 --list-formats-ext
```

```
root@ATK-IMX6U:~# v4l2-ctl --device=/dev/video1 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUYV-16

        Index       : 1
        Type        : Video Capture
        Pixel Format: 'UYVY'
        Name        : UYVY-16

        Index       : 2
        Type        : Video Capture
        Pixel Format: 'RGBP'
        Name        : RGB565_LE

        Index       : 3
        Type        : Video Capture
        Pixel Format: 'RGBP'
        Name        : RGB565_BE

root@ATK-IMX6U:~#
```

图 3.16.3 3 查看驱动中正点原子 ov7725 摄像头支持的格式

3.16.3.1 使用 gstreamer 采集

➢ 采集图像并显示

可以设置采集分辨率 640x320 与 320x240 分辨率采集。由于 pdf 格式问题，复制时分段复制，这里 **format=(string)RGB16** 即 RGB565 采集，可支持 **format=(string)YUY2** 即 YUYV 格式采集。（注意，由于最初正点原子摄像头 ov7725 是设计给单片机竖屏使用的，所以在横屏的 I.MX6U 中可能看到的图像是 90 度旋转的，这并不是驱动的问题，我们可以将摄像头侧着拍摄，图像则转正，在使用效果上是没有任何影响的，采集的图像也非常清晰，虽然只有 640x480 分辨率的大小）。

```
gst-launch-1.0 -v v4l2src device=/dev/video1 ! "video/x-raw, format=(string)RGB16, width=(int)640, height=(int)480" ! imxv4l2sink
```

```
root@ATK-IMX6U:~# gst-launch-1.0 -v v4l2src device=/dev/video1 ! "video/x-raw, format=(string)RGB16, width=(int)640, height=(int)480" ! imxv4l2sink
====== IMXV4L2SINK: 4.1.6 build on Nov 17 2020 14:36:18. ======
Setting pipeline to PAUSED ...
display(/dev/fb0) resolution is (480x272).
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
/GstPipeline:pipeline0/GstV4l2Src:v4l2src0.GstPad:src: caps = "video/x-raw\,\ format\=\(string\)RGB16\,\ framerate\=\(fraction\)100/1\,\ width\=\(int\)640\,\ height\=\(int\)480\,\ interlace-mode\=\(string\)progressive\,\ pixel-aspect-ratio\=\(fraction\)1/1\,\ colorimetry\=\(string\)sRGB"
/GstPipeline:pipeline0/GstCapsFilter:capsfilter0.GstPad:src: caps = "video/x-raw\,\ format\=\(string\)RGB16\,\ framerate\=\(fraction\)100/1\,\ width\=\(int\)640\,\ height\=\(int\)480\,\ interlace-mode\=\(string\)progressive\,\ pixel-aspect-ratio\=\(fraction\)1/1\,\ colorimetry\=\(string\)sRGB"
/GstPipeline:pipeline0/GstImxV4l2Sink:imxv4l2sink0.GstPad:sink: caps = "video/x-raw\,\ format\=\(string\)RGB16\,\ framerate\=\(fraction\)100/1\,\ width\=\(int\)640\,\ height\=\(int\)480\,\ interlace-mode\=\(string\)progressive\,\ pixel-aspect-ratio\=\(fraction\)1/1\,\ colorimetry\=\(string\)sRGB"
/GstPipeline:pipeline0/GstCapsFilter:capsfilter0.GstPad:sink: caps = "video/x-raw\,\ format\=\(string\)RGB16\,\ framerate\=\(fraction\)100/1\,\ width\=\(int\)640\,\ height\=\(int\)480\,\ interlace-mode\=\(string\)progressive\,\ pixel-aspect-ratio\=\(fraction\)1/1\,\ colorimetry\=\(string\)sRGB"
v4l2sink need allocate 3 buffers.
```

图 3.16.3.1 2 使用 gstreamer 采集

效果图略，十分流畅。

拍照功能等请参考上上一节 ov5640 的相关指令。

3.16.3.2 使用 ov772x_camera 采集

Ov772x_camera 应用程序是正点原子编写的一个用于测试的程序。在串口终端直接执行 ov772x_camera 即可。

```
ov772x_camera
```

```
root@ATK-IMX6U:~# ov772x_camera

Brief: ov772x camera Application
Version: version 1.0
Date: 2021.01.30
Author: www.openedv.com
```

图 3.16.3.2 1 使用正点原子编写的 ov772x_camera 程序采集

效果图略，十分流畅。

3.17 USB 摄像头测试

实验前请准备 USB 摄像头，符合 UVC（USB video device class）协议的摄像头均可。UVC，全称为：USB video class 或 USB video device class，是 Microsoft 与另外几家设备厂商联合推出的为 USB 视频捕获设备定义的协议标准。符合 UVC 规格的硬件设备在不需要安装任何的驱动程序下即可在主机中正常使用。

插上 USB 摄像头到 USB 接口处，USB 设备驱动打印如下信息

```
root@ATK-IMX6U:~# [  115.792682] usb 2-1.4: new high-speed USB device number 4 using ci_hdrc
[  116.027710] uvcvideo: Found UVC 1.00 device USB Camera (0fde:2024)
[  116.051065] input: USB Camera as /devices/platform/soc/2100000.aips-bus/2184200.usb/ci_hdrc.1/usb2/2-1/2-1.4/2-1.4:1.0/input/input3
[  116.066241] usbcore: registered new interface driver uvcvideo
[  116.072555] USB Video Class driver (1.1.1)
[  116.136960] usbcore: registered new interface driver snd-usb-audio

root@ATK-IMX6U:~#
```

图 3.17 1 USB 摄像头打印的信息

查看设备节点，video1 是 ov5640/ov7725/ov2640 节点，video2 是 USB 摄像头节点。（注这里 video1 不一定是 ov5640 节点，这与 ov5640 驱动和 USB 摄像头它们驱动的加载顺序相关）。

```
ls /dev/video*
```

```
root@ATK-IMX6U:~# ls /dev/video*
/dev/video0  /dev/video1  /dev/video2
root@ATK-IMX6U:~#
```

图 3.17 2 查看 USB 摄像头的节点

查看摄像头支持格式、分辨率及帧率。本次 video1 是 ov5640 设备节点，video2 是 USB 摄像头的节点。

```
v4l2-ctl --device=/dev/video2 --list-formats-ext
```

```
root@ATK-IMX6U:~# v4l2-ctl --device=/dev/video2 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'MJPG' (compressed)
        Name        : MJPEG
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 424x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x360
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 960x540
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)

        Index       : 1
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUV 4:2:2 (YUYV)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 160x120
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 176x144
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 352x288
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 424x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x360
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 960x540
                        Interval: Discrete 0.100s (10.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.100s (10.000 fps)

root@ATK-IMX6U:~#
```

图 3.17 3 查看 USB 摄像头支持的格式

3.17.1 使用 gstreamer 采集

由上图可知看到 USB 摄像头支持两种格式采集，一种是“YUYV”别外一种是“MJPG”。由于使用 gstreamer 使用元件 imxv4l2src（可用 v4l2src）不能设置 MJPG 格式采集，它只能设置 YUYV 格式采集。传入字符参数是 format=(string)YUY2。

本次采集的图像大小为 640*480 30fps，如果需要设置其他分辨率采集需要严格按照上面的参数。执行下面指令开始采集，并显示到 LCD 上面，按“Ctrl + c”终止指令。

```
gst-launch-1.0 -v imxv4l2src device=/dev/video2 ! "video/x-raw, format=(string)YUY2, width=(int)640, height=(int)480, framerate=(fraction)30/1" ! imxv4l2sink
```

演示效果图略。

使用 usb 摄像头保存录像，请参考如下指令，注意 usb 摄像头对应的节点名称。录像完成后在当前目录保存了一个 video.mkv 文件，使用 gst-play-1.0 或者 gplay-1.0 直接播放即可。

```
gst-launch-1.0 -vvv -e v4l2src num-buffers=1000 device=/dev/video2 ! image/jpeg,width=640,height=480,framerate=30/1,rate=30 ! matroskamux ! filesink location=video.mkv
```

3.17.2 使用 ffmpeg 采集

```
ffmpeg -t 10 -s 320x240 -pix_fmt yuyv422 -i /dev/video2 video.avi
```

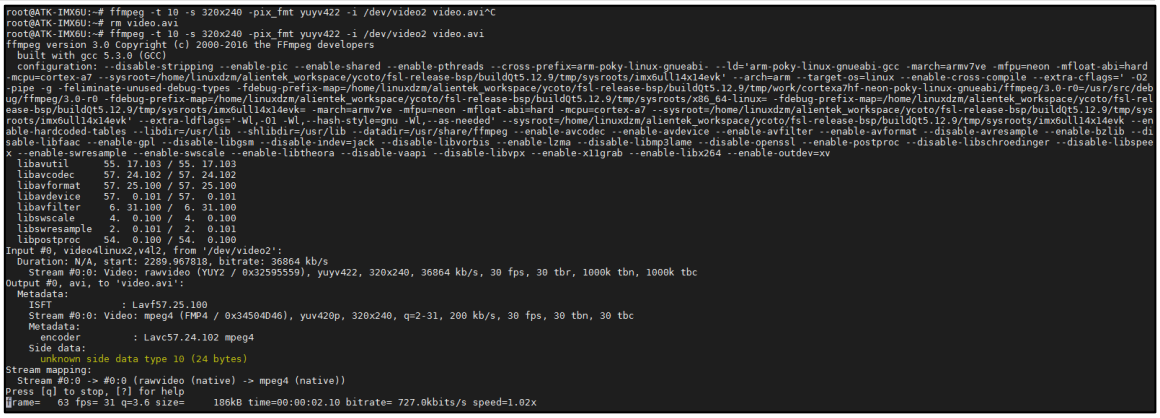


图 3.17.2 1 使用 ffmpeg 保存 USB 摄像头录像保存视频

指令解释：

（1） ffmpeg: ffmpeg 指令

（2） -t 10: 设置采集时间为 10s

（3） -s 340x240: 设置采集的分辨率大小，与 ov5640 不一样，USB 摄像头可以使用 ffmpeg 设置采集的分辨率大小。采集的图像分辨率越大，可能采集时性能不足。

（4） -pix_fmt yuyv422: 设置采集的格式，支持(YUYV)yuyv422。

（5） -i /dev/video2: 指定 USB 摄像头设备节点。

（6） video.avi: 输出文件名，不同的格式会自动选择不同的编码，目前测试 avi 格式流畅。

使用 ffmpeg 设置编码指令请执行 ffmpeg -help 查看相关帮助与用法，本手册不再详细说明。

3.18 EC20 4G 模块上网测试

ALPHA	MINI
本实验支持	本实验不支持/但是可使用 PCIE 转 USB 座子接 4G 模块

移远模块自带了一套驱动和拨号软件，驱动叫 GobiNet，我们发布的内核源码已经含 GobiNet 驱动。移远 EC20 4G 模块驱动已经编译成模块，可以直接插上 EC20 4G 模块使用。也可以不使用移远的驱动 GobiNet，直接使用 ppp 拨号上网亦可。

WWAN LED 指示灯说明，当为低的时候 LED 灯点亮，参考电路如下：

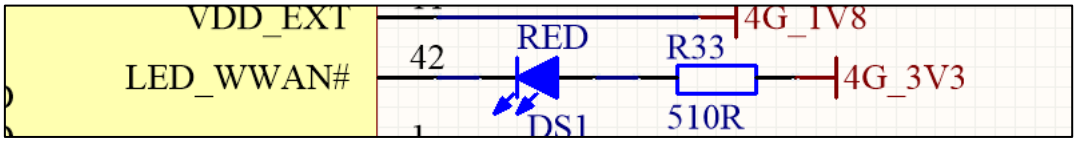


图 3.18 1 WWWAN LED 指示灯

默认状态下 LED_WWAN 对应的 LED 灯闪烁情况：

引脚工作状态	所指示的网络状态
慢闪(200ms 高/1800ms 低)	找网状态
慢闪(1800ms 高/200ms 低)	待机状态
快闪(125ms 高/125ms 低)	数据传输模式
高电平	通话中

实验前准备：

- EC20 4G 模块
- 4 G 上网卡
- 天线（用于放大信号）

正点原子 ALPHA 底板上预留 4G 模块接口，ME3630-W，EC20 等 4G 模块的安装。准备 EC20 模块，请自行在网上购买，注意购买时需要买天线，单单模块是不能正常工作的！（备注：EC20 有许多类型模块，目前测试过的是 EC20-CE 模块，其中 EC20-CE 系列又有多种模块，不同的模块功能不一样，比如支持的运营商不一样，详细请咨询卖家），其他 EC20 系列请自行测试，理论上驱动一样，有需求找移远技术支持。）。将 EC20 4G 模块插到 4G 模块接口处，拧上螺丝。保证 4G 模块与座子接口吻合连接。请使用原装天线，把天线连接到 4G 模块的 MAIN 接口处。

正确插入 4G 卡（支持的运营商，请咨询对应模块的卖家，注意有些可能模块不支持物联网卡，请使用普通 4G 卡测试）及插好模块，开发板启动后底板上的 WWAN LED 会亮绿灯。如果 WWAN LED 绿灯未亮起，请检查模块是否正确连接插入，4G 卡是否插入，天线是否接好，开发板是必须插上配带的 12V 电源，不能只用串口 USB_TTL 供电。

进行 4G 模块测试前，将 4G 卡插到底板的 SIM 卡槽里，再插上 EC20 4G 模块，同时插上天线，天线接到模块的 MAIN 处。正确插入 4G 卡与天线后，开发板启动后底板上的 WWAN LED 会亮绿灯，若此灯不亮，请检查 4G 卡是否插对位置，天线是否连接正确，再重插模块试试。必须插上开发板使用的电源！否则供电不足，模块无法正常工作。模块安装如下图所示：

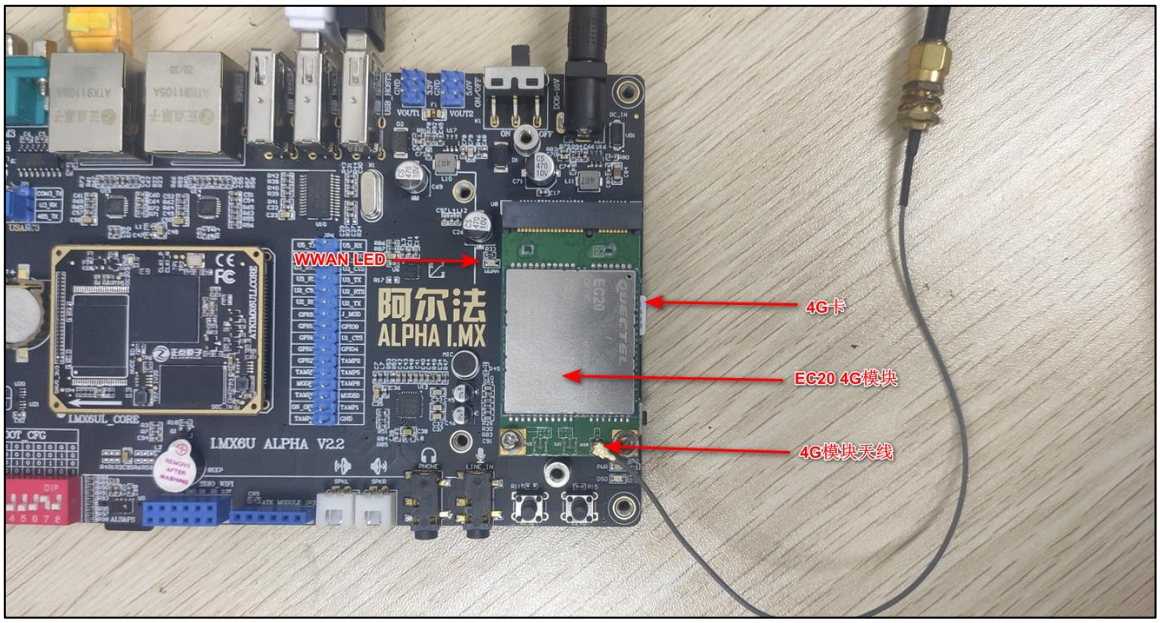


图 3.18 2 EC20 连接示意图

开发板开机启动打印 EC20 4G 模块信息如下：

```
    9.350514] GobiNet: Quectel_WCDMA&LTE_Linux&Android_GobiNet_Driver_V1.3.0
    9.445199] GobiNet 2-1.2:1.4 eth2: register 'GobiNet' at usb-ci_hdrc.1-1.2, GobiNet Ethernet Device, e2:31:c3:f2:0b:8f
    9.518012] creating qcqmi2
    9.523654] usbcore: registered new interface driver GobiNet
    9.549558] usbcore: registered new interface driver usbserial
    9.569008] usbcore: registered new interface driver usbserial_generic
    9.604294] usbserial: USB Serial support registered for generic
    9.657961] usbcore: registered new interface driver option
    9.669406] usbserial: USB Serial support registered for GSM modem (1-port)
    9.697945] idVendor=2c7c, idProduct=125, bInterfaceNumber =0
    9.711742] option 2-1.2:1.0: GSM modem (1-port) converter detected
    9.738500] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB0
    9.766128] idVendor=2c7c, idProduct=125, bInterfaceNumber =1
    9.773203] option 2-1.2:1.1: GSM modem (1-port) converter detected
    9.788844] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB1
    9.804162] idVendor=2c7c, idProduct=125, bInterfaceNumber =2
ALSA: Restoring mixer settings...[    9.822896] option 2-1.2:1.2: GSM modem (1-port) converter detected

[    9.847321] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB2
[    9.862980] idVendor=2c7c, idProduct=125, bInterfaceNumber =3
[    9.884866] option 2-1.2:1.3: GSM modem (1-port) converter detected
[    9.904102] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB3
No state is present for card Camera
```

图 3.18 3 EC20 打印的 USB 信息

查看是否存在/dev/qcqmil2 节点，如果存在的话就说明 GobiNet 驱动成功，如下图所示：

```
ls /dev/q*
```

```
root@ATK-IMX6U:~# ls /dev/q*
/dev/qcqmi2
root@ATK-IMX6U:~#
```

图 3.18 4 查看 EC20 生成的节点

再查看是否生成/dev/ttyUSB0~3 节点

```
ls /dev/ttyUSB*
```

```
root@ATK-IMX6U:~# ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB1  /dev/ttyUSB2  /dev/ttyUSB3
root@ATK-IMX6U:~# _
```

图 3.18 5 生成的 USB 节点

这四路 ttyUSB 的功能如下图图 3.18. 6 所示，不全部测试这些功能了，这里我们只测试上网功能。详细请自行参考 EC20 4G 模块手册。

Product			USB Driver	Interface
UC15	VID: 0x05c6	PID: 0x9090	USB Serial	ttyUSB0 → DM
UC20	VID: 0x05c6	PID: 0x9003		
EC25	VID: 0x2c7c	PID: 0x0125		ttyUSB1 → For GPS NMEA message output
EC21	VID: 0x2c7c	PID: 0x0121		ttyUSB2 → For AT commands
EC20	VID: 0x05c6	PID: 0x9215		
EC20 R2.0	VID: 0x2c7c	PID: 0x0125		ttyUSB3 → For PPP connections or AT commands
EG91	VID: 0x2c7c	PID: 0x0191		
EG95	VID: 0x2c7c	PID: 0x0195		
UC20	VID: 0x05c6	PID: 0x9003	GobiNet or QMI WWAN	ethX or wwanX → Interface 4 can be used as USB Network Adapter
EC25	VID: 0x2c7c	PID: 0x0125		
EC21	VID: 0x2c7c	PID: 0x0121		
EC20	VID: 0x05c6	PID: 0x9215		
EC20 R2.0	VID: 0x2c7c	PID: 0x0125		
EG91	VID: 0x2c7c	PID: 0x0191		
EG95	VID: 0x2c7c	PID: 0x0195		
EG06	VID: 0x2c7c	PID: 0x0306		
EP06	VID: 0x2c7c	PID: 0x0306		
EM06	VID: 0x2c7c	PID: 0x0306		
BG96	VID: 0x2c7c	PID: 0x0296		

图 3.18 6 四路 ttyUSB 的功能示意图

3.18.1 ppp 拨号上网

进入/home/root/shell/4G 目录下，这个目录存放着测试 4G 模块的脚本，如果您没看见 4G 这个目录，请回到 2.2 小节下载最新的固件更新。

```
cd /home/root/shell/4G
ls
```

4G 测试脚本如下。

```
root@ATK-IMX6U:~/shell/4G# ls
disconnect    ecm-on-10010  gosuncn_ecm_dialer        gosuncn_ecm_dialer_10010  gosuncn_options          gosuncn_ppp_dialer_10010  ppp-on-10000  ppp-on-10086
ecm-on-10000  ecm-on-10086  gosuncn_ecm_dialer_10000  gosuncn_ecm_dialer_10086  gosuncn_ppp_dialer_10000  gosuncn_ppp_dialer_10086  ppp-on-10010
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.1 1 查看 4G 测试脚本

脚本解释：

ppp 拨号主要是 ppp-on-1000、ppp-on-10010 和 ppp-on-10086。这三个脚本分别是不同的运营商配置的 APN 值不一样。ppp-on-1000、ppp-on-10010 和 ppp-on-10086 分别是电信卡需要执行的脚本、联通卡需要执行的脚本和移动卡需要执行的脚本。

比如本次测试使用的是电信卡，那么执行的脚本是 ppp-on-10000。

```
./ppp-on-10000 &                    // &的作用是放到后台运行
```

```
root@ATK-IMX6U:~/shell/4G# ./ppp-on-10000 &
[1] 798
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.1 2 执行电信卡运行的 ppp 上网测试脚本

```
pppd options in effect:
debug           # (from gosuncn_options)
nodetach                # (from gosuncn_options)
persist         # (from gosuncn_options)
dump            # (from gosuncn_options)
noauth          # (from gosuncn_options)
user Anyname            # (from gosuncn_options)
password ??????         # (from gosuncn_options)
/dev/ttyUSB2            # (from gosuncn_options)
115200          # (from gosuncn_options)
lock            # (from gosuncn_options)
connect chat -v -f /home/root/shell/4G/gosuncn_ppp_dialer_10000         # (from command line)
crtscts         # (from gosuncn_options)
modem           # (from gosuncn_options)
novj            # (from gosuncn_options)
ipcp-accept-local               # (from gosuncn_options)
ipcp-accept-remote              # (from gosuncn_options)
noipdefault             # (from gosuncn_options)
defaultroute            # (from gosuncn_options)
usepeerdns              # (from gosuncn_options)
noccp           # (from gosuncn_options)
nobsdcomp               # (from gosuncn_options)
ATE
OK
ATH
OK
ATP
OK
AT+CGDCONT=1,"IP","CTNET"
OK
ATD*99#
CONNECT
Script chat -v -f /home/root/shell/4G/gosuncn_ppp_dialer_10000 finished (pid 810), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x3e53d37a> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x0 <asyncmap 0x0> <auth pap> <magic 0x506a13f2> <pcomp> <accomp>]
sent [LCP ConfAck id=0x0 <asyncmap 0x0> <auth pap> <magic 0x506a13f2> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x3e53d37a> <pcomp> <accomp>]
sent [PAP AuthReq id=0x1 user="Anyname" password=<hidden>]
rcvd [LCP DiscReq id=0x1 magic=0x506a13f2]
rcvd [PAP AuthAck id=0x1 ""]
PAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x0]
sent [IPCP ConfNak id=0x0 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 10.15.92.94> <ms-dns1 202.96.128.86> <ms-dns2 202.96.134.133>]
sent [IPCP ConfReq id=0x2 <addr 10.15.92.94> <ms-dns1 202.96.128.86> <ms-dns2 202.96.134.133>]
rcvd [IPCP ConfReq id=0x1]
sent [IPCP ConfAck id=0x1]
rcvd [IPCP ConfAck id=0x2 <addr 10.15.92.94> <ms-dns1 202.96.128.86> <ms-dns2 202.96.134.133>]
Could not determine remote IP address: defaulting to 10.64.64.64
not replacing default route to eth0 [192.168.1.1]
local  IP address 10.15.92.94
remote IP address 10.64.64.64
primary   DNS address 202.96.128.86
secondary DNS address 202.96.134.133
Script /etc/ppp/ip-up started (pid 819)
```

获取到本地IP与远程网关IP

图 3.18.1 3 获取 IP 成功

如果开发板同时插上了网线，因为此系统默认会只让一个网卡连外网，设置网关为 4G 模块的路由规则。使用 route 指令查看路由表。

```
route
```

因为本人已经插上网线，上网会优先选择 eth0/eth1。如果用户没有插网线，就不需要添加路由表啦，因为你的网卡只有 4G 网卡上网，系统就会选择 4G 网卡上网。

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
10.64.64.64     *               255.255.255.255 UH    0      0        0 ppp0
public1.114dns. 192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth0
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.1 4 默认是 eth0 上网

我们需要先添加 4G 模块的网关地址，然后删除默认的 eth0 的网关地址，再使用 route 指令查看添加是否成功

```
route add default gw 10.64.64.64
route del default gw 192.168.1.1
route
```

可以看到下面 default 项，已经修改为 ppp0 上网。

```
root@ATK-IMX6U:~/shell/4G# route add default gw 10.64.64.64
root@ATK-IMX6U:~/shell/4G# route del default gw 192.168.1.1
root@ATK-IMX6U:~/shell/4G# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.64.64.64     0.0.0.0         UG    0      0        0 ppp0
10.64.64.64     *               255.255.255.255 UH    0      0        0 ppp0
public1.114dns. 192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth0
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.1 5 修改为 ppp0 上网

使用 ifconfig 指令查看获取的 ip 地址，表明 4G 网络可以与 eth0/eth1 共存，上网通过切换默认的路由表来控制连外网即可！

```
ifconfig
```

```
root@ATK-IMX6U:~/shell/4G# ifconfig
eth0      Link encap:Ethernet  HWaddr 88:89:3c:f1:8a:ba
          inet addr:192.168.1.124  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::8a89:3cff:fef1:8aba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3113 errors:0 dropped:24 overruns:0 frame:0
          TX packets:96 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:389540 (380.4 KiB)  TX bytes:12983 (12.6 KiB)

eth1      Link encap:Ethernet  HWaddr 88:ff:c4:7a:e8:9b
          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)

eth2      Link encap:Ethernet  HWaddr f2:20:cd:2b:5e:53
          UP BROADCAST NOARP 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)

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:42 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3745 (3.6 KiB)  TX bytes:3745 (3.6 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.15.92.94  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:517 (517.0 B)  TX bytes:272 (272.0 B)

root@ATK-IMX6U:~/shell/4G#
```


图 3.18.1 6 eth 网络与 4G 网络共存

通过 ping www.baidu.com 来测试是否能上网。-I 参数是指定 ppp0(4G 网络)，按“Ctrl +c”结束 ping。看到下图结果表明能上网。

```
ping www.baidu.com -I ppp0
```

```
root@ATK-IMX6U:~/shell/4G# ping www.baidu.com -I ppp0
PING www.a.shifen.com (14.215.177.38) from 10.15.92.94 ppp0: 56(84) bytes of data.
64 bytes from 14.215.177.38: icmp_seq=1 ttl=54 time=46.6 ms
64 bytes from 14.215.177.38: icmp_seq=2 ttl=54 time=35.1 ms
64 bytes from 14.215.177.38: icmp_seq=3 ttl=54 time=53.9 ms
^C
--- www.a.shifen.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 35.101/45.224/53.962/7.762 ms
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.1 7 ping 百度测试

3.18.2 使用 quectel-CM

使用 quectel-CM 拨号程序工具（这个工具是我们预先交叉编译好放进文件系统/usr/sbin 目录下面的），方便用户使用。此工具在我们 I.MX6U 嵌入式 Linux 开发指南有讲到。

如果已经做了上面的 ppp 拨号，那么我们需要断开，执行下面的指令。

```
./disconnect
```

```
root@ATK-IMX6U:~/shell/4G# ./disconnect
Terminating on signal 15
Connect time 14.2 minutes.
Sent 741 bytes, received 1413 bytes.
root@ATK-IMX6U:~/shell/4G# Script /etc/ppp/ip-down started (pid 838)
sent [LCP TermReq id=0x2 "User request"]
rcvd [LCP TermAck id=0x2]
Connection terminated.
Script /etc/ppp/ip-down finished (pid 838), status = 0x0

[1]+  Done(5)                 ./ppp-on-10000
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.2 1 断开上一步的 pppd 拨号连接

查看该工具的用法，执行下面的指令。

```
quectel-CM -h
```

```
root@ATK-IMX6U:~/shell/4G# quectel-CM -h
[08-04_16:49:37:047] Usage: quectel-CM [-s [apn [user password auth]]] [-p pincode] [-f logfilename]
[08-04_16:49:37:049] -s [apn [user password auth]] Set apn/user/password/auth get from your network provider
[08-04_16:49:37:049] -p pincode                    Verify sim card pin if sim card is locked
[08-04_16:49:37:049] -f logfilename                Save log message of this program to file
[08-04_16:49:37:049] Example 1: quectel-CM
[08-04_16:49:37:049] Example 2: quectel-CM -s 3gnet
[08-04_16:49:37:049] Example 3: quectel-CM -s 3gnet carl 1234 0 -p 1234 -f gobinet_log.txt
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.2 2 查看 quectel-CM 用法

可以看到-s 参数是指定 apn 类型，移动卡 apn 一般是 cmnet，联通卡 apn 一般是 3gnet，电信卡一般是 ctnet。备注：APN 指一种网络接入技术，通常是通过手机上网时必须配置的一个参数，它决定了手机通过哪种接入方式来访问网络。

```
quectel-CM &                    // 如果不清楚 apn 类型，可以直接输入 quectel-CM
```

```
root@ATK-IMX6U:~/shell/4G# quectel-CM &
[1] 852
root@ATK-IMX6U:~/shell/4G# [08-04_16:55:53:075] WCDMA&LTE_QConnectManager_Linux&Android_V1.1.34
[08-04_16:55:53:076] quectel-CM profile[1] = (null)/(null)/(null)/0, pincode = (null)
[08-04_16:55:53:077] Find /sys/bus/usb/devices/2-1.2 idVendor=2c7c idProduct=0125
[08-04_16:55:53:077] Find /sys/bus/usb/devices/2-1.2:1.4/net/eth2
[08-04_16:55:53:077] Find usbnet_adapter = eth2
[08-04_16:55:53:078] Find /sys/bus/usb/devices/2-1.2:1.4/GobiQMI/qcqmi2
[08-04_16:55:53:078] Find qmichannel = /dev/qcqmi2
[08-04_16:55:53:110] Get clientWDS = 7
[08-04_16:55:53:141] Get clientDMS = 8
[08-04_16:55:53:173] Get clientNAS = 9
[08-04_16:55:53:205] Get clientUIM = 10
[08-04_16:55:53:238] Get clientWDA = 11
[08-04_16:55:53:270] requestBaseBandVersion EC20CEFAGR06A10M4G
[08-04_16:55:53:366] requestGetSIMStatus SIMStatus: SIM_READY
[08-04_16:55:53:398] requestGetProfile[1] CTNET///0
[08-04_16:55:53:430] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[08-04_16:55:53:462] requestQueryDataCall IPv4ConnectionStatus: DISCONNECTED
[08-04_16:55:53:526] requestRegistrationState2 MCC: 460, MNC: 11, PS: Attached, DataCap: LTE
[08-04_16:55:53:590] requestSetupDataCall WdsConnectionIPv4Handle: 0x873b2580
[  961.161981] IPv6: ADDRCONF(NETDEV_CHANGE): eth2: link becomes ready
[08-04_16:55:53:685] requestQueryDataCall IPv4ConnectionStatus: CONNECTED
[08-04_16:55:53:719] ifconfig eth2 up
[08-04_16:55:53:774] busybox udhcpc -f -n -q -t 5 -i eth2
[08-04_16:55:53:822] udhcpc (v1.24.1) started
[  961.520776] GobiNet 2-1.2:1.4 eth2: kevent 12 may have been dropped
[08-04_16:55:54:065] Sending discover...
[08-04_16:55:54:125] Sending select for 10.15.92.94...
[08-04_16:55:54:186] Lease of 10.15.92.94 obtained, lease time 7200
[08-04_16:55:54:352] /etc/udhcpc.d/50default: Adding DNS 202.96.128.86
[08-04_16:55:54:352] /etc/udhcpc.d/50default: Adding DNS 202.96.134.133

root@ATK-IMX6U:~/shell/4G#
```

获取到的IP地址

图 3.18.2 3 上网连接，获取 IP

查看网卡信息，使用 ifconfig 指令，ALPHA 开发板底板默认有两个网卡，已经使用 eth0 和 eth1 设备名，那么 eth2 就是 4G 网卡。可以看到 quectel-CM 获取了 ip 10.15.92.94。

```
ifconfig
```

```
root@ATK-IMX6U:~/shell/4G# ifconfig
eth0      Link encap:Ethernet  HWaddr 88:89:3c:f1:8a:ba
          inet addr:192.168.1.124  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::8a89:3cff:fef1:8aba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12059 errors:0 dropped:87 overruns:0 frame:0
          TX packets:112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1393785 (1.3 MiB)  TX bytes:15607 (15.2 KiB)

eth1      Link encap:Ethernet  HWaddr 88:ff:c4:7a:e8:9b
          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)

eth2      Link encap:Ethernet  HWaddr f2:20:cd:2b:5e:53
          inet addr:10.15.92.94  Bcast:10.15.92.95  Mask:255.255.255.252
          UP BROADCAST RUNNING NOARP MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:15 errors:0 dropped:0 overruns:0 frame:0
          TX packets:47 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2389 (2.3 KiB)  TX bytes:7395 (7.2 KiB)

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:42 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3745 (3.6 KiB)  TX bytes:3745 (3.6 KiB)

root@ATK-IMX6U:~/shell/4G#
```

图 3.18.2 4 查看 eth2 获取的 IP

因为编者已经插上网线，在上一步已经把 eth0 默认的路由表删除，所以无需再删除 eth0 路由表，现在默认就是选择 eth2 上网了。

执行下面的指令测试上网，参数“-I”指定网卡。这里要指定 eth2 网卡。可按“Ctrl + c”终止 ping 指令。出现如下结果，说明 ping 百度成功。

```
ping www.baidu.com -I eth2
```

```
root@ATK-IMX6U:~/shell/4G# ping www.baidu.com -I eth2
PING www.a.shifen.com (14.215.177.38) from 10.15.92.94 eth2: 56(84) bytes of data.
64 bytes from 14.215.177.38: icmp_seq=1 ttl=54 time=40.8 ms
64 bytes from 14.215.177.38: icmp_seq=2 ttl=54 time=31.2 ms
64 bytes from 14.215.177.38: icmp_seq=3 ttl=54 time=58.4 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3004ms
rtt min/avg/max/mdev = 31.243/43.539/58.478/11.275 ms
root@ATK-IMX6U:~/shell/4G#
```

图 3.18.2 5 ping 百度测试

3.19 视频播放测试

文件系统含 GStreamer（流媒体应用的开源多媒体框架），它采用基于插件（plugin）和管道（pipeline）体系结构，提供了 GStreamer 相关的库，同时提供了相关的应用程序。简单来说，用户可以把它当作一种多媒体播放器。然后我们就可以使用播放器播放我们的音频和视频了，但 GStreamer 的功能远远不仅仅是这些。

由于 CPU 没有硬件多媒体解码器且 CPU 资源有限，播放视频分辨率不能太高，帧率也不要太高，否则播放会有卡顿感。

播放视频测试，执行下面指令，播放系统目录/meida/test_movie.avi，分辨率为 856x480，25 帧的视频文件。按“Ctrl + c”结束播放。

```
gst-play-1.0 /media/test_movie.avi
```

```
root@ATK-IMX6U:/# gst-play-1.0 /media/test_movie.avi
Can't get caps from capture device, use the default setting.
Perhaps haven't capture device.
Press 'k' to see a list of keyboard shortcuts.
Now playing /media/test_movie.avi
Prerolling...
====== AIUR: 4.1.6 build on Nov 11 2020 22:22:29. ======
       Core: AVI_PARSER_03.05.28  build on Oct 10 2016 07:21:58
 file: /usr/lib/imx-mm/parser/lib_avi_parser_arm11_elinux.so.3.1
-----------------------
    Track 00 [video_0] Enabled
        Duration: 0:00:20.040000000
        Language: und
    Mime:
        video/mpeg, systemstream=(boolean)false, parsed=(boolean)true, mpegversion=(int)4, width=(int)856, height=(int)480, framerate=(fraction)25/1
-----------------------
====== IMXV4L2SINK: 4.1.6 build on Nov 11 2020 22:22:51. ======
display(/dev/fb0) resolution is (800x480).
-----------------------
    Track 01 [audio_0] Enabled
        Duration: 0:00:20.062040000
        Language: und
    Mime:
        audio/mpeg, mpegversion=(int)1, channels=(int)2, rate=(int)44100, bitrate=(int)0
-----------------------
Home directory not accessible: Permission denied
Redistribute latency...
Home directory not accessible: Permission denied

====== BEEP: 4.1.6 build on Nov 11 2020 22:22:37. ======
       Core: MP3 decoder Wrapper  build on Mar 21 2014 15:04:50
 file: /usr/lib/imx-mm/audio-codec/wrap/lib_mp3d_wrap_arm12_elinux.so.3
CODEC: BLN_MAD-MMCODECS_MP3D_ARM_02.13.00_CORTEX-A8  build on Jul 12 2016 13:15:30.
Redistribute latency...
v4l2sink need allocate 3 buffers.
0:00:02.4 / 0:00:20.0
root@ATK-IMX6U:/# _
```

图 3.19 1 执行指令播放视频

LCD 屏幕上播放的视频：

图 3.19 2 LCD 屏幕上播放的视频

3.20 AP3216C 测试

ALPHA	MINI
本实验支持	本实验不支持

AP3216C 简介：

ALPHA 开发板上通过 I2C1 连接了一个三合一环境传感器：AP3216C，AP3216C 是由敦南可以推出的一款传感器，其支持环境光强度(ALS)、接近距离(PS)和红外线强度(IR)这三个环境参数检测。AP3216C 的特点如下：

①、I2C 接口，快速模式下波特率可以到 400Kbit/S

②、多种工作模式选择：ALS、PS+IR、ALS+PS+IR、PD 等等。
③、内建温度补偿电路。
④、宽工作温度范围(-30° C ~ +80° C)。
⑤、超小封装，4.1mm x 2.4mm x 1.35mm
⑥、环境光传感器具有 16 为分辨率。
⑦、接近传感器和红外传感器具有 10 为分辨率。

AP3216C 常被用于手机、平板、导航设备等，其内置的接近传感器可以用于检测是否有物体接近，比如手机上用来检测耳朵是否接触听筒，如果检测到的话就表示正在打电话，手机就会关闭手机屏幕以省电。也可以使用环境光传感器检测光照强度，可以实现自动背光亮度调节。

进入开发板文件系统执行下面指令读取环境传感器的环境参数值，根据开发板所处环境不同，环境参数值不同，先用下面指令读取一次环境参数值，再用手接近 AP3216C 传感器（ALPHA 底板 U8 处），再用指令读取相应的参数值，参数值会有比较大的变化。

读取环境光强度值（ALS）

```
cat /sys/class/misc/ap3216c/als
```

```
root@ATK-IMX6U:~# cat /sys/class/misc/ap3216c/als
1065
root@ATK-IMX6U:~#
```

图 3.20 1 读取环境光强度值

读取接近距离(PS)

```
cat /sys/class/misc/ap3216c/ps
```

```
root@ATK-IMX6U:~# cat /sys/class/misc/ap3216c/ps
0
root@ATK-IMX6U:~#
```

图 3.20 2 读取接近距离值

```
cat /sys/class/misc/ap3216c/ir
```

读取红外线强度(IR)

```
root@ATK-IMX6U:~# cat /sys/class/misc/ap3216c/ir
4
root@ATK-IMX6U:~#
```

图 3.20 3 读取红外线强度值

3.21 icm20608 测试

ALPHA	MINI
本实验支持	本实验不支持

ICM-20608 简介：

ICM-20608 是 InvenSense 出品的一款 6 轴 MEMS 传感器，包括 3 轴加速度和 3 轴陀螺仪。

ICM-20608 尺寸非常小，只有 3x3x0.75mm，采用 16P 的 LGA 封装。ICM-20608 内部有一个 512 字节的 FIFO。陀螺仪的量程范围可以编程设置，可选择±250，±500，±1000 和±2000°/s，加速度的量程范围也可以编程设置，可选择±2g，±4g，±4g，±8g 和±16g。陀螺仪和加速度计都是 16 位的 ADC，并且支持 I2C 和 SPI 两种协议，使用 I2C 接口的话通信速度最高可以达到 400KHz，使用 SPI 接口的话通信速度最高可达到 8MHz。I.MX6U-ALPHA 开发板上的 ICM-20608 通过 SPI 接口和 I.MX6U 连接在一起。ICM-20608 特性如下：

①、陀螺仪支持 X，Y 和 Z 三轴输出，内部集成 16 位 ADC，测量范围可设置：±250，±

500，±1000 和±2000°/s。

②、加速度计支持 X，Y 和 Z 轴输出，内部集成 16 位 ADC，测量范围可设置：±2g，±4g，±4g，±8g 和±16g。

③、用户可编程中断。

④、内部包含 512 字节的 FIFO。

⑤、内部包含一个数字温度传感器。

⑥、耐 10000g 的冲击。

⑦、支持快速 I2C，速度可达 400KHz。

⑧、支持 SPI，速度可达 8MHz。

I.MX6U-ALPHA 使用 SPI3 接口连接了一个六轴传感器 ICM-20608，由正点原子提供 linux 驱动程序与用户测试程序。

```
cd /home/root/driver/icm20608/
ls
```

```
root@ATK-IMX6U:~# cd /home/root/driver/icm20608/
root@ATK-IMX6U:~/driver/icm20608# ls
icm20608.ko  icm20608App
root@ATK-IMX6U:~/driver/icm20608#
```

图 3.21 1 拷贝驱动文件与程序程序到文件系统

使用 insmod 指令安装驱动文件 icm20608.ko，安装成功如下图。驱动在安装时会读取 ICM-20608 的 ID 信息，读取返回 0XAE 表明驱动正常。

```
insmod icm20608.ko     // 开机 Qt GUI 会默认加这个驱动，可不用执行这项
```

```
root@ATK-IMX6U:~/driver/icm20608# insmod icm20608.ko
[  592.736137] ICM20608 ID = 0XAE
root@ATK-IMX6U:~/driver/icm20608#
```

图 3.21 2 安装驱动文件

同时在/dev/生成 icm20608 节点。

```
ls /dev/icm20608
```

```
root@ATK-IMX6U:~/driver/icm20608# ls /dev/icm20608
/dev/icm20608
root@ATK-IMX6U:~/driver/icm20608#
```

图 3.21 3 查看/dev 下生成的 icm20608 节点

执行下面的指令可获取 6 轴传感器的值，应用程序会从/dev/icm20608 节点不断获取并打印数据信息。

```
./icm20608App /dev/icm20608
```

```
root@ATK-IMX6U:~/driver/icm20608# ./icm20608App /dev/icm20608

原始值:
gx = -14, gy = 19, gz = 3
ax = 2, ay = 0, az = 2112
temp = 4548
实际值:act gx = -0.85°/S, act gy = 1.16°/S, act gz = 0.18°/S
act ax = 0.00g, act ay = 0.00g, act az = 1.03g
act temp = 38.84°C

原始值:
gx = -14, gy = 19, gz = 3
ax = 1, ay = -2, az = 2105
temp = 4546
实际值:act gx = -0.85°/S, act gy = 1.16°/S, act gz = 0.18°/S
act ax = 0.00g, act ay = -0.00g, act az = 1.03g
act temp = 38.83°C
```

图 3.21 4 执行指令获取内部数据

3.22 USB WIFI 模块测试

本实验使用 USB WIFI RTL8188EUS/RTL8188CUS 模块（正点原子购买 ALPHA 开发板赠送的 USB WIFI 模块，可直接插到电脑上联网测试好坏，若需要安装 USB WIFI 驱动则需要安

装驱动精灵安装相应的驱动即可！），使用 USB 2.0 HOST 接口，ALPHA 底板 4 个 USB 接口都可以。正点原子提供 USB WIFI 测试脚本 alientek_usb_wifi_setup.sh，仅供用户参考。

测试前准备 USB WIFI RTL8188EUS/RTL8188CUS 模块，一般 USB 设备都是可带电插拔的，同理我们的 RTL8188EUS/RTL8188CUS 模块也是支持热插拔的，可在系统起来后再插上 USB WIFI 模块。测试前请插上 12v 电源！

为测试 USB WIFI 正点原子已经编写了一个脚本“alientek_usb_wifi_setup.sh”脚本内容仅供参考，默认把它放在/home/root/shell/wifi 目录下。如下图。

```
cd /home/root/shell/wifi
ls
```

```
root@ATK-IMX6U:~# cd /home/root/shell/wifi
root@ATK-IMX6U:~/shell/wifi# ls
alientek_usb_wifi_setup.sh
root@ATK-IMX6U:~/shell/wifi#
```

图 3.22 1 拷贝 WIFI 测试脚本到文件系统目录

开发板在 USB 接口处插上 USB WIFI RTL8188EUS 模块，如下图（下图为 ALPHA 底板）。

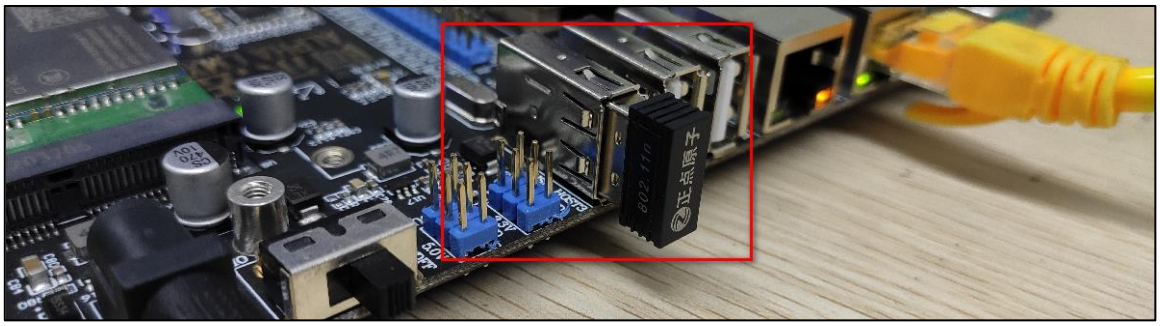


图 3.22 2 USB WIF 连接示意图

开机启动系统后插上 USB WIFI，驱动打印信息如下：

```
root@ATK-IMX6U:~/shell/wifi# [ 2089.162780] usb 2-1.4: new high-speed USB device number 5 using ci_hdrc
[ 2089.361262] RTL871X: module init start
[ 2089.367948] RTL871X: rtl8188eu v4.3.0.9_15178.20150907
[ 2089.378748] bFWReady == _FALSE call reset 8051...
[ 2089.415072] RTL871X: rtw_ndev_init(wlan0)
[ 2089.431449] usbcore: registered new interface driver rtl8188eu
[ 2089.439625] RTL871X: module init ret=0
[ 2089.973522] ==> rtl8188e_iol_efuse_patch
[ 2090.369065] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 2090.376189] RTL871X: set bssid:00:00:00:00:00:00
[ 2090.381303] RTL871X: set ssid [g▒isQ▒J▒) .▒▒▒▒F|▒T▒vZ.c3▒>▒▒▒▒▒▒] fw_state=0x00000008
[ 2091.699907] RTL871X: indicate disassoc
[ 2091.713794] RTL871X: set ssid [@PHICOMM_94] fw_state=0x00000008
[ 2091.720916] RTL871X: set bssid:d8:c8:e9:f7:41:96
[ 2092.083334] RTL871X: start auth
[ 2092.689284] RTL871X: auth success, start assoc
[ 2092.732925] RTL871X: assoc success
[ 2092.736590] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 2092.762009] RsvdPageNum: 8

root@ATK-IMX6U:~/shell/wifi#
```

图 3.22 3 打印的驱动信息

查看 USB WIFI 的网卡信息，使用 ifconfig 指令，如下图示，wlan0 是 USB WIFI 的节点。

```
ifconfig wlan0 up        // 若默认没打开 wlan0 则需要执行此项。
ifconfig
```

```
root@ATK-IMX6U:~# ifconfig
eth0      Link encap:Ethernet  HWaddr b6:9d:81:5e:04:81
          inet addr:192.168.1.182  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::b49d:81ff:fe5e:481/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:580 errors:0 dropped:4 overruns:0 frame:0
          TX packets:73 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:61399 (59.9 KiB)  TX bytes:11190 (10.9 KiB)

eth1      Link encap:Ethernet  HWaddr 62:e5:14:bc:c3:aa
          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)

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:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2623 (2.5 KiB)  TX bytes:2623 (2.5 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:13:ef:f1:0b:23
          inet6 addr: fe80::213:efff:fef1:b23/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:2474 (2.4 KiB)

root@ATK-IMX6U:~#
```

图 3.22 4 查看网卡信息

3.22.1 Station（上网）模式

本次实验目的：使用 USB WIFI 连接无线网络并测试网络是否能上网。
扫描附近无线网络信息并打印，如下图扫描到本公司的无线网络名称“ALIENTEK-YF”，还能看到加密类型等。（备注：还可以用其他指令来扫描无线网络信息，如 iwlist wlan0 scan）

```
wpa_cli -i wlan0 scan_result        // 此指令需要启动进入后再插 WIFI 才生效，需要 wpa_supplicant 在运行。
```

```
root@ATK-IMX6U:~/shell/wifi# wpa_cli -i wlan0 scan_result
bssid / frequency / signal level / flags / ssid
e4:0e:ee:f2:11:15       2412    72      [WPA2-PSK-CCMP][ESS]
88:f8:72:8d:f3:19       2462    56      [WPA2-PSK-CCMP][ESS]
88:f8:72:8d:f3:18       2462    49      [WPA2-PSK-CCMP][WPS][ESS]       ZZK
e4:0e:ee:f2:11:19       2412    47      [WPA2-PSK-CCMP][ESS]
e4:0e:ee:f2:11:14       2412    43      [WPA2-PSK-CCMP][WPS][ESS]       ALIENTEK-YF
60:3a:7c:ff:d6:21       2462    42      [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      1DW2.4g
0c:b5:27:99:ff:ac       2412    47      [WPA2-PSK-CCMP][WPS][ESS]       ALIENTEK-FAE
08:40:f3:ff:25:c1       2432    44      [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] ALIENTEK-HYS
0c:b5:27:99:ff:ad       2412    44      [WPA2-PSK-CCMP][WPS][ESS]
9c:a6:15:d0:b2:7e       2437    25      [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    hqfs2.4g
c4:ad:34:fc:8e:86       2447    23      [WPA-PSK-CCMP][WPA2-PSK-CCMP][WPS][ESS] Admin2.4G
9c:6f:52:ad:2f:f8       2437    23      [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS]    CU_XbfG
e8:c4:17:c1:1b:0d       2462    23      [WPA-PSK-CCMP][WPA2-PSK-CCMP][ESS]      ChinaNet-JqHm
d8:c8:e9:f7:41:96       2417    23      [ESS]   @PHICOMM_94
5c:0e:8b:8b:22:41       2412    23      [ESS]   Wireless_BY
root@ATK-IMX6U:~/shell/wifi#
```

搜索到“ALIENTEK-YF”热点名

图 3.22.1 1 扫描附近的无线网络信息

```
source ./alientek_usb_wifi_setup.sh -m station -i ALIENTEK-YF  -p 1590202**** -d wlan0
```

参数解释：
-m station ： 设置成 station 模式
-i ALIENTEK-YF ： 无线网络名称(ssid)。
-p 1590202**** ： 无线网络密码(psk)。
-d wlan0 ： USB WIFI 节点

看到下图已经获取到 ip 信息就代表连接无线网络成功，如果没有获取到 ip 信息，请检查密码是否正确或者重试指令。

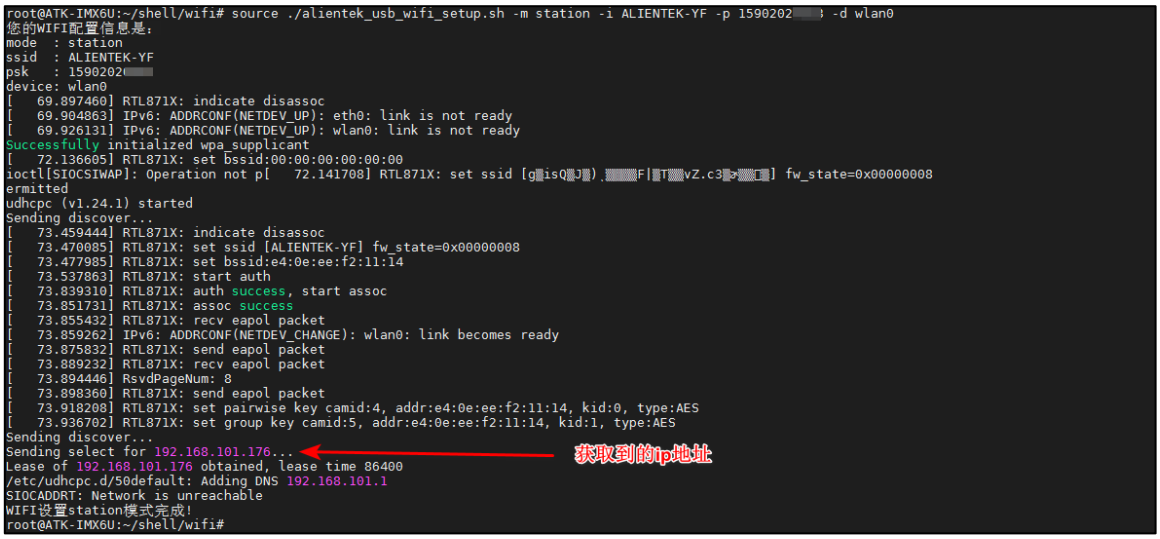


图 3.22.1 2 执行指令连接无线网络

若未能正常获取 ip，等待 RTL871X: set group key camid:5 这句话出现后输入 udhcpc -i wlan0 指令重新获取 ip。若没有这句话，请检查无线网络信息是否正确。

测试是否能上网，使用 ping 指令 ping 百度，可按 ctrl+c 终止执行指令

```
ping www.baidu.com -I wlan0
```

看到如下信息，就代表能上网

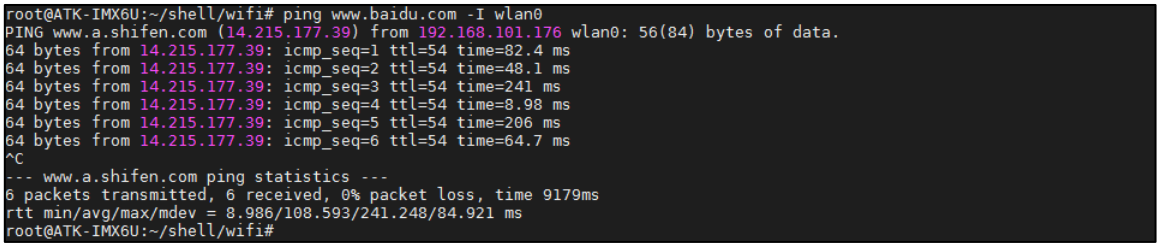


图 3.22.1 3 ping 通百度测试

3.22.2 SoftAP（热点）模式

本次实验目的：测试 USB WIFI 开启热点，与手机连接

```
source ./alientek_usb_wifi_setup.sh -m softap -d wlan0
```

```
root@ATK-IMX6U:~/shell/wifi# source ./alientek_usb_wifi_setup.sh -m softap -d wlan0
您的WIFI配置信息是:
mode   : softap
device: wlan0
[    49.893738] ==> rtl8188e_iol_efuse_patch
[    50.291367] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
udhcpd (v1.24.1) started
Configuration file: //home/root/shell/wifi/wifi.conf
drv->ifindex=6
l2_sock_recv==l2_sock_xmit=0x0x1f7e8e8
+rtl871x_sta_deauth_ops, ff:ff:ff:ff:ff:ff is deauth, reason=2
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
Using interface wlan0 with hwaddr 00:13:ef:f4:26:dd and ssid "alientek_softap"
rtl871x_set_wps_assoc_resp_ie
rtl871x_set_wps_beacon_ie
rtl871x_set_wps_probe_resp_ie
random: Only 11/20 bytes of strong random data available from /dev/random
random: Not enough entropy pool available for secure operations
WPA: Not enough entropy in random pool for secure operat[    52.675534] RTL871X: assoc success
ions - update keys later when the first station connects
rtl871x[    52.682817] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
_set_beacon_ops
rtl871x_set_hidden_ssid_ops
ioctl[RTL_IOCTL_HOSTAPD]: Invalid argument
rtl871x_set_key_ops
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
[    52.709505] RTL871X: set group key camid:1, addr:00:00:00:00:00:00, kid:1, type:TKIP
WIFI设置softap模式完成!
```

图 3.22.2 1 把 USB WIFI 设置成热点模式

打开手机设置，可以看到 USB WIFI 发出的热点，名称为“alientek_softap”，密码默认为 12345678。输入密码后点击连接，即可连接到无线热点。

图 3.22.2 2 手机识别 USB WIFI 热点名

图 3.22.2 4 输入密码并连接

注意：请更新到最新的文件系统（v2.3 版本文件系统以上），直接输入密码即可连接，不会出现连接上了又断开的情况。

v2.2 版本及以下的文件系统会出现连接上了又断开的情况，用户请将连接 alientek_softap 热点信息设置为静态获取设置 IP 模式。如下图。注因手机厂商不一样，界面设置的方法可能不一样

图 3.22.2 5 设置静态 IP

可以看到手机已经连接了热点，这样手机与开发板的 WIFI 构成了一个无线局域网。

图 3.22.2 6 手机成功连接 USB WIFI 热点

如需要修改热点名称与热点密码，用户可以编辑脚本内容设置个人的热点信息。下图是拷贝 alientek_usb_wifi_setup.sh 脚本到 Windows 上查看。

```
126  #softap Mode（热点模式）
127 if [ "$mode" == "softap" ]; then
128     rfkill unblock all
129     processkill
130     sleep 1
131     sync
132     echo -ne "interface=wlan0\nssid=alientek_softap\ndriver=rtl871xdrv\nchannel=6\nhw_mode=g\nignore_broadcast_ssid=0\n
133 auth_algs=1\nwpa=3\nwpa_passphrase=12345678\nwpa_key_mgmt=WPA-PSK\nwpa_pairwise=TKIP\nrsn_pairwise=CCMP" > $wifi_conf
134     a=$(ifconfig |grep -E "br0" | grep -v grep | awk '{print $0}')
135     if [ -n "$a" ];then
136         brctl delif br0 $device
137         brctl delif br0 $device
138         ifconfig br0 down
139     fi
```

图 3.22.2 7 查看脚本配置的热点名称及密码

3.22.3 Bridge（桥接）模式

本次实验目的：测试 USB WIFI 开启热点，并桥接到有线网络，让手机连接到热点并上网。

ALPHA 开发板的网口请将网线插上其中一个或者两个，并且确保有线网络能上网。

用 ifconfig 查看有线网络 eth0 或者 eth1 能否获取 ip，并测试 eth0 是否能上网。下图是将网线插到开发板底板 ENET2 接口处，节点是 eth0，如果插到 ENET1 接口，节点是 eth1。MINI 底板只有一个网口，节点是 eth0。

```
ifconfig
```

```
root@ATK-IMX6U:~/shell/wifi# ifconfig
eth0      Link encap:Ethernet  HWaddr d6:f7:32:82:a4:c7
          inet addr:192.168.1.125  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::d4f7:32ff:fe82:a4c7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3223 errors:0 dropped:53 overruns:0 frame:0
          TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:257007 (250.9 KiB)  TX bytes:13570 (13.2 KiB)

eth1      Link encap:Ethernet  HWaddr fa:5a:53:54:73:4a
          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)

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:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2381 (2.3 KiB)  TX bytes:2381 (2.3 KiB)

root@ATK-IMX6U:~/shell/wifi#
```

图 3.22.3 1 查看有线网络的节点

source ./alientek_usb_wifi_setup.sh -m bridge -d wlan0 -e eth0

-e eth0 ：桥接的有线网络节点，根据实际情况桥接到对应的网络节点 eth0/eth1。

```
root@ATK-IMX6U:~/shell/wifi# source ./alientek_usb_wifi_setup.sh -m bridge -d wlan0 -e eth0
您的WIFI配置信息是:
mode  : bridge
device: wlan0
ethernet  : eth0
[  644.373786] ==> rtl8188e_iol_efuse_patch
[  644.838264] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[  645.073453] fec 20b4000.ethernet eth0: Freescale FEC PHY driver [SMSC LAN8710/LAN8720] (mii_bus:phy_addr=20b4000.ethernet:01, irq=-1)
[  647.230565] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[  648.153332] fec 20b4000.ethernet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
[  648.161223] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
udhcpd: /etc/udhcpd.conf: No such file or directory
udhcpd (v1.24.1) started
[  648.636671] device eth0 entered promiscuous mode
[  648.672517] device wlan0 entered promiscuous mode
[  648.704700] br0: port 1(eth0) entered forwarding state
[  648.709931] br0: port 1(eth0) entered forwarding state
Configuration file: //home/root/shell/wifi/wifi.conf
drv->ifindex=6
+rtl871x_sta_deauth_ops, ff:ff:ff:ff:ff:ff is deauth, reason=2
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
rtl871x_set_key_ops
Using interface wlan0 with hwaddr 00:13:ef:f1:0b:23 and ssid 'alientek_bridge'
rtl871x_set_wps_assoc_resp_ie
rtl871x_set_wps_beacon_ie
rtl871x_set_wps_probe_resp_ie
rtl871x_set_key_ops
rtl871x_set[  648.987534] RTL871X: set group key camid:1, addr:00:00:00:00:00:00, kid:1, type:TKIP
_beacon_ops
[  649.014064] RTL871X: assoc success
[  649.018367] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
rtl871x_set_hidden_ssid ignore_br[  649.028654] br0: port 2(wlan0) entered forwarding state
oadcast_ssid:0, alientek_bridge,1[  649.034177] br0: port 2(wlan0) entered forwarding state
5
rtl871x_set_acl
WIFI设置bridge模式完成!
root@ATK-IMX6U:~/shell/wifi#
```

图 3.22.3 2 执行指令将无线网络 wlan0 桥接到 eth0

打开手机设置，可以看到 USB WIFI 发出的热点，名称为“alientek_bridge”，密码默认为 12345678。输入密码后点击连接，即可连接到无线热点。并测试能否上网，打开手机的浏览器测试即可。

图 3.22.3 3 输入密码，连接成功示意图

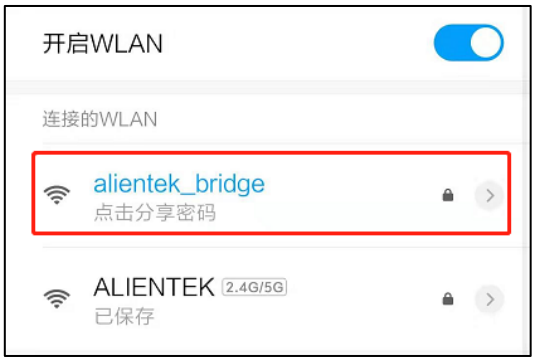


图 3.22.3 4 输入密码，连接成功示意图

如需要修改热点名称与热点密码，用户可以拷贝该脚本到 Windows 下编辑脚本内容设置个人的热点信息。

```
97  #bridge Mode(桥接模式)
98  if [ "$mode" == "bridge" ]; then
99      rfkill unblock all
100     ifconfig $device down
101     ifconfig $device up
102     ifconfig $ethernet down
103     ifconfig $ethernet up
104     sleep 2
105     if [ "$WIFI_MODE" == "station" ]; then
106         wpa_supplicant -B -D wext -i $device -c /etc/wpa_supplicant.conf
107     fi
108     processkill
109     sleep 1
110     sync
111     echo -ne "interface=wlan0\nssid=alientek_bridge\ndriver=rtl871xdrv\nchannel=6\nhw_mode=g\nignore_broadcast_ssid=0\n
112 auth_algs=1\nwpa=3\nwpa_passphrase=12345678\nwpa_key_mgmt=WPA-PSK\nwpa_pairwise=TKIP\nrsn_pairwise=CCMP\nbridge=br0" > $wifi_conf
113     rm -rf /var/lib/misc/*
114     touch /var/lib/misc/udhcpd.leases
115     udhcpd -fS /etc/udhcpd.conf &
116     ifconfig $device 0.0.0.0
117     brctl addbr br0
```

图 3.22.2 3 脚本设置的热点名称与密码

3.23 ME3630-W 4G 模块测试

ALPHA	MINI
本实验支持	本实验不支持/但是可使用 PCIE 转 USB 座子接 4G 模块

实验前准备 ME3630-W 4G 模块（正点原子店铺有售卖）、天线和一张上网卡（电信 4G 卡、联通 4G 卡或者移动 4G 卡）。

进行 4G 模块测试前，将移动或者联通 4G 卡插到底板的 SIM 卡槽里，再插上 ME3630-W 4G 模块，同时插上天线，天线接到模块的 MAIN 处。正确插入 4G 卡与天线后，正常加载驱动后，开发板启动后底板上的 WWAN LED 会亮绿灯（WWAN LED 指示灯说明参考 3.18 小节），若此灯不亮，请检查 4G 卡是否插好（卡没插好也不会亮绿灯），ME3630-W 是否插稳，天线是否连接正确，如测试失败多数是没有正确连接模块。必须插上开发板使用的电源！否则供电不足，模块无法正常工作 ME3630-W 4G 模块安装如下图所示：

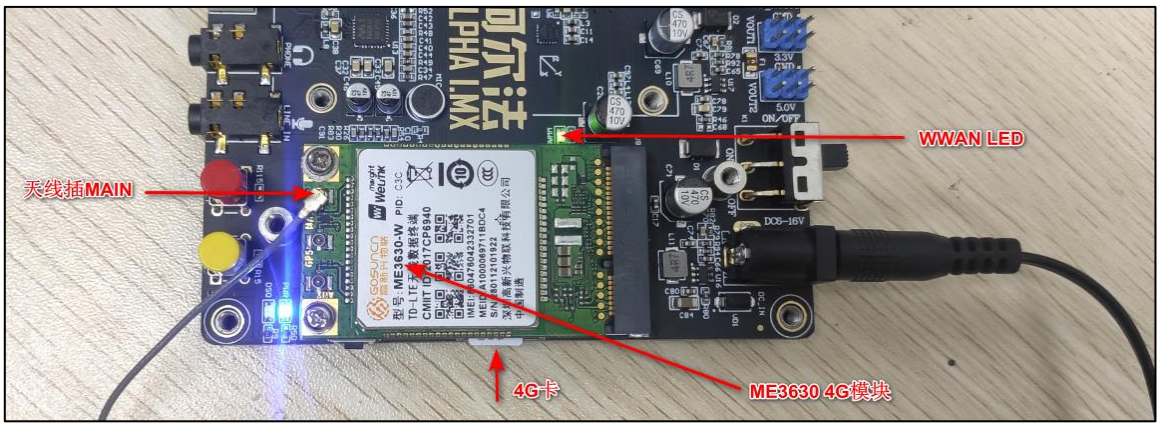


图 3.23 1 ME3630 模块连接示意图

开机启动串口终端打印 4G 模块加载成功，并生成 ttyUSB0~ ttyUSB2。

```
[   11.257163] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB0
[   11.265943] idVendor=19d2, idProduct=1476, bInterfaceNumber =1
[   11.273157] option 2-1.2:1.1: GSM modem (1-port) converter detected
[   11.287559] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB1
NFS daemon support not enabled in kernel
[   11.303909] idVendor=19d2, idProduct=1476, bInterfaceNumber =2
[   11.310052] option 2-1.2:1.2: GSM modem (1-port) converter detected
[   11.326015] usb 2-1.2: GSM modem (1-port) converter now attached to ttyUSB2
```

图 3.23 2 串口终端打印加载 4G 模块的相关信息

进入/home/root/shell/4G 目录下，这个目录存放着测试 4G 模块的脚本，如果您没看见 4G 这个目录，请回到 2.2 小节下载最新的固件更新。

```
cd /home/root/shell/4G
ls
```

4G 测试脚本如下，（下图文件系统版本需要 v2.4 以上）。

```
root@ATK-IMX6U:~/shell/4G# ls
disconnect      gosuncn_options           gosuncn_ppp_dialer_10086  ppp-on-10086
ECM_DEMO        gosuncn_ppp_dialer_10000  ppp-on-10000
ECM_DEMO_AUTO   gosuncn_ppp_dialer_10010  ppp-on-10010
root@ATK-IMX6U:~/shell/4G#
```

图 3.23 3 查看测试 4G 模块的相关脚本

脚本解释：

ppp 拨号主要是 ppp-on-1000、ppp-on-10010 和 ppp-on-10086。这三个脚本分别是不同的运营商配置的 APN 值不一样。ppp-on-1000、ppp-on-10010 和 ppp-on-10086 分别是电信卡需要执行的脚本、联通卡需要执行的脚本和移动卡需要执行的脚本。

ECM 接口主要用 ECM_DEMO 和 ECM_DEMO_AUTO 二进制执行文件。

3.23.1 pppd 拨号上网

比如本次测试使用的是电信卡，那么执行的脚本是 ppp-on-10000。

```
./ppp-on-10000 &                    // &的作用是放到后台运行
```

```
root@ATK-IMX6U:~/shell/4G# ./ppp-on-10000 &
[1] 798
root@ATK-IMX6U:~/shell/4G#
```

图 3.23.1 1 执行脚本进行拨号上网

```
root@ATK-IMX6U:~/shell/4G#
pppd options in effect:
debug           # (from gosuncn_options)
nodetach                # (from gosuncn_options)
persist         # (from gosuncn_options)
dump            # (from gosuncn_options)
noauth          # (from gosuncn_options)
user Anyname            # (from gosuncn_options)
password ??????         # (from gosuncn_options)
/dev/ttyUSB2            # (from gosuncn_options)
115200          # (from gosuncn_options)
lock            # (from gosuncn_options)
connect chat -v -f /home/root/shell/4G/gosuncn_ppp_dialer_10000         # (from command line)
crtscts         # (from gosuncn_options)
modem           # (from gosuncn_options)
novj            # (from gosuncn_options)
ipcp-accept-local               # (from gosuncn_options)
ipcp-accept-remote              # (from gosuncn_options)
noipdefault             # (from gosuncn_options)
defaultroute            # (from gosuncn_options)
usepeerdns              # (from gosuncn_options)
noccp           # (from gosuncn_options)
nobsdcomp               # (from gosuncn_options)
ATE
OK
ATH
OK
ATP
OK
AT+CGDCONT=1,"IP","CTNET"
OK
ATD*99#
CONNECT
Script chat -v -f /home/root/shell/4G/gosuncn_ppp_dialer_10000 finished (pid 811), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x1d82625b> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x9ffc0b94> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x9ffc0b94> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x1d82625b> <pcomp> <accomp>]
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x1 <addr 10.28.103.254>]
sent [IPCP ConfAck id=0x1 <addr 10.28.103.254>]
rcvd [IPV6CP ConfReq id=0x1 <addr fe80::2952:0bda:a762:8c48>]
Unsupported protocol 'IPv6 Control Protocol' (0x8057) received
sent [LCP ProtRej id=0x2 80 57 01 01 00 0e 01 0a 29 52 0b da a7 62 8c 48]
rcvd [IPCP ConfNak id=0x1 <addr 10.28.103.253> <ms-dns1 202.96.134.33> <ms-dns2 202.96.134.33>]
sent [IPCP ConfReq id=0x2 <addr 10.28.103.253> <ms-dns1 202.96.134.33> <ms-dns2 202.96.134.33>]
rcvd [IPCP ConfAck id=0x2 <addr 10.28.103.253> <ms-dns1 202.96.134.33> <ms-dns2 202.96.134.33>]
not replacing default route to eth0 [192.168.1.1]
local  IP address 10.28.103.253
remote IP address 10.28.103.254
primary   DNS address 202.96.134.33
secondary DNS address 202.96.134.33
Script /etc/ppp/ip-up started (pid 820)
Script /etc/ppp/ip-up finished (pid 820), status = 0x0
```

本地IP与远程网关IP

图 3.23.1 2 拨号连接成功打印的信息

如果开发板同时插上了网线，因为此系统默认会只让一个网卡连外网，设置网关为 4G 模块的路由规则。使用 route 指令查看路由表。

```
route
```

因为本人已经插上网线，上网会优先选择 eth0/eth1。如果用户没有插网线，就不需要添加路由表啦，因为你的网卡只有 4G 网卡上网，系统就会选择 4G 网卡上网。

```
root@ATK-IMX6U:~/shell/4G# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth0
10.28.103.254   *               255.255.255.255 UH    0      0        0 ppp0
public1.114dns. 192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth0
root@ATK-IMX6U:~/shell/4G#
```

图 3.23.1 3 查看路由表，默认是 eth0 上网

我们需要先添加 4G 模块的网关地址，然后删除默认的 eth0 的网关地址，再使用 route 指令查看添加是否成功

```
route add default gw 10.28.103.254
route del default gw 192.168.1.1
route
```

可以看到下面 default 项，已经修改为 ppp0 上网。

```
root@ATK-IMX6U:~/shell/4G# route add default gw 10.28.103.254
root@ATK-IMX6U:~/shell/4G# route del default gw 192.168.1.1
root@ATK-IMX6U:~/shell/4G# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.28.103.254   0.0.0.0         UG    0      0        0 ppp0
10.28.103.254   *               255.255.255.255 UH    0      0        0 ppp0
public1.114dns. 192.168.1.1     255.255.255.255 UGH   0      0        0 eth0
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth0
root@ATK-IMX6U:~/shell/4G#
```

图 3.23.1 4 修改路由表，修改为 ppp0 上网

使用 ifconfig 指令查看获取的 ip 地址，表明 4G 网络可以与 eth0/eth1 共存，上网通过切换默认的路由表来控制连外网即可！

```
ifconfig
```

```
root@ATK-IMX6U:~/shell/4G# ifconfig
eth0      Link encap:Ethernet  HWaddr 88:89:3c:f1:8a:ba
          inet addr:192.168.1.124  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::8a89:3cff:fef1:8aba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12336 errors:0 dropped:81 overruns:0 frame:0
          TX packets:123 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1498747 (1.4 MiB)  TX bytes:17729 (17.3 KiB)

eth1      Link encap:Ethernet  HWaddr 88:ff:c4:7a:e8:9b
          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)

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:42 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3745 (3.6 KiB)  TX bytes:3745 (3.6 KiB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.28.103.253  P-t-P:10.28.103.254  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:535 (535.0 B)  TX bytes:270 (270.0 B)

usb0      Link encap:Ethernet  HWaddr 7a:29:05:56:d5:e7
          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)

root@ATK-IMX6U:~/shell/4G#
```


图 3.23.1 5 查看获取的 ip 地址

通过 ping www.baidu.com 来测试是否能上网。-I 参数是指定 ppp0(4G 网络)，按“Ctrl +c”结束 ping。看到下图结果表明能上网。

```
ping www.baidu.com -I ppp0
```

```
root@ATK-IMX6U:~/shell/4G# ping www.baidu.com -I ppp0
PING www.a.shifen.com (14.215.177.39) from 10.28.103.253 ppp0: 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=53 time=34.4 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=53 time=46.7 ms
64 bytes from 14.215.177.39: icmp_seq=3 ttl=53 time=31.5 ms
64 bytes from 14.215.177.39: icmp_seq=4 ttl=53 time=52.3 ms
^C
--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 31.534/41.265/52.314/8.568 ms
root@ATK-IMX6U:~/shell/4G#
```

图 3.23.1 6 ping 百度测试

3.23.2 通过 ECM 上网

在操作系统看来，CDC ECM 设备就是一个虚拟以太网卡，包含标准网卡需要的 MAC 地址和 IP 地址。

ECM_DEMO_AUTO 和 ECM_DEMO 是高新兴 ECM 上网的程序，指令简介如下：

```
ECM_DEMO -t up              //开启 ecm 上网
ECM_DEMO -t down            //关闭 ecm 上网
ECM_DEMO -t up -p /dev/ttyUSB1 -a 3gnet   //开启的同时指定对应的 apn 和拨号端口
ECM_DEMO -t down -p /dev/ttyUSB1          //关闭 ecm 上网
```

详细的参数可以查看源代码或者执行 ECM_DEMO -h 查看

ECM_DEMO_AUTO 和 ECM_DEMO 参数是一样的，区别是 ECM_DEMO 执行完流程就会退出，ECM_DEMO_AUTO 会一直运行并每隔一段时间会检查是否断网，断网了会自动重连 ECM_DEMO_AUTO 默认会自动执行拨号流程，故适合做开机自启的程序。

要配置 ECM 模式上网，如果运行了 pppd 上网，请先执行 disconnect 脚本断开 pppd 拨号上网，再执行下面的指令配置成 ECM 模式链接网络。

```
./disconnect
./ECM_DEMO -t up
```

```
root@ATK-IMX6U:~/shell/4G#      ./ECM_DEMO -t up
[  199.742172] IPv6: ADDRCONF(NETDEV_CHANGE): usb0: link becomes ready
[  199.748540] cdc_ether 2-1.2:1.3 usb0: kevent 12 may have been dropped
[11_01_19:19:41] ECM_DEMO success
```

图 3.23.2 1 断开 ppp-on 上网，执行 ecm-on 上网

使用 ifconfig 指令查看获取的 ip 地址，如果没有获取到 ip 地址使用 udhcpc -i usb0 获取。

```
ifconfig
```

```
root@ATK-IMX6U:~/shell/4G# ifconfig
eth0      Link encap:Ethernet  HWaddr 88:89:3c:f1:8a:ba
          inet addr:192.168.1.124  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::8a89:3cff:fef1:8aba/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17001 errors:0 dropped:108 overruns:0 frame:0
          TX packets:132 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2039926 (1.9 MiB)  TX bytes:18875 (18.4 KiB)

eth1      Link encap:Ethernet  HWaddr 88:ff:c4:7a:e8:9b
          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)

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:42 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3745 (3.6 KiB)  TX bytes:3745 (3.6 KiB)

usb0      Link encap:Ethernet  HWaddr 7a:29:05:56:d5:e7
          inet addr:10.28.103.253  Bcast:10.28.103.255  Mask:255.255.255.252
          inet6 addr: fe80::7829:5ff:fe56:d5e7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:17 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:1938 (1.8 KiB)  TX bytes:7079 (6.9 KiB)

root@ATK-IMX6U:~/shell/4G#
```

图 3.23.2 2 查看获取的 ip 地址

通过 ping www.baidu.com 来测试是否能上网。-I 参数是指定 usb0(4G 网络)，按“Ctrl +c”结束

```
ping www.baidu.com -I usb0      //  “-I”参数是指定网卡名
```

```
root@ATK-IMX6U:~/shell/4G# ping www.baidu.com -I usb0
PING www.baidu.com (14.215.177.39) from 10.28.103.253 usb0: 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=53 time=37.7 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=53 time=86.7 ms
64 bytes from 14.215.177.39: icmp_seq=3 ttl=53 time=34.6 ms
64 bytes from 14.215.177.39: icmp_seq=4 ttl=53 time=63.2 ms
^C
--- www.baidu.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 34.633/55.602/86.715/21.124 ms
root@ATK-IMX6U:~/shell/4G#
```

图 3.23.2 3 ping 百度测试上网

3.24 SDIO WIFI 测试

实验前准备正点原子 SDIO WIFI 模块（RTL8189），非赠送，可以在正点原子官方淘宝店铺购买。

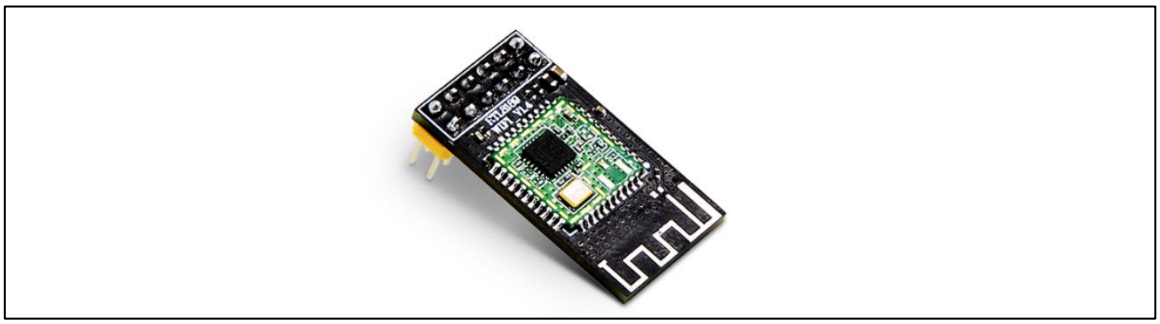


图 3.24 1 正点原子 SDIO WIFI 模块

由于底板 SDIO WIFI 接口与 TF 卡槽接口共用了大部分管脚，所以在使用 SDIO WIFI 时不能使用 TF 卡。也就是说，我们测试 SDIO WIFI 时不能从 TF 卡启动系统。我们应该从 eMMC 或者 Nand Flash 启动系统。注意，不要同时插入 SDIO WIFI 与 USB WIFI。因为测试脚本里默认写的是 wlan0，只能是其中一个 WIFI。

本次实验我们从 eMMC 启动系统，启动系统前，请将 SDIO WIFI 模块插进 SDIO 接口处，如下图所示：

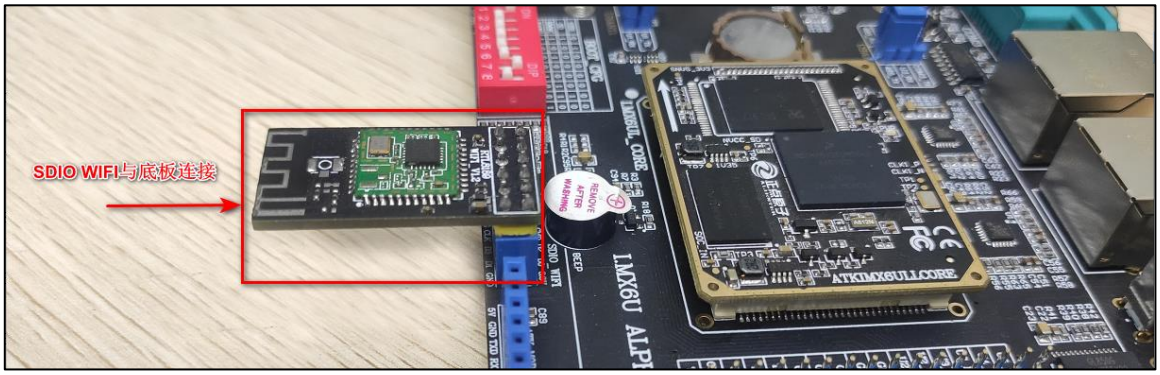


图 3.24 2 SDIO WIFI 接法示意图

RTL8189 的驱动文件出厂默认放在/home/root/driver/rtl8189/目录下，执行下成的指令查看。

```
cd /home/root/driver/rtl8189/
ls
```

```
root@ATK-IMX6U:~# cd /home/root/driver/rtl8189/
root@ATK-IMX6U:~/driver/rtl8189# ls
8189fs.ko
root@ATK-IMX6U:~/driver/rtl8189#
```

图 3.24 3 拷贝驱动文件到文件系统目录下

为了方便用户测试，我们简化用户的测试步骤，，编写了一个初始化 SDIO WIFI 的脚本。放在/home/root/shell/wifi 下，脚本名为 alientek_sdio_wifi_init.sh。脚本内容和解释如下。此脚本是后面新版本文件系统添加的，请更新文件系统到 v2.3 及以上。

```
#!/bin/sh
#安装驱动模块
insmod /home/root/driver/rtl8189/8189fs.ko
#解锁射频
rfkill unblock all
#杀死后台运行的程序
killall wpa_supplicant
```

我们直接执行直接进入/home/root/shell/wifi，执行此脚本初始化 SDIO WIFI。注意，使用 SDIO WIFI 前需要执行此脚本！

```
cd /home/root/shell/wifi
```

```
root@ATK-IMX6U:~/shell/wifi# ./alientek_sdio_wifi_init.sh
[ 1147.528219] RTL871X: module init start
[ 1147.532001] RTL871X: rtl8189fs v4.3.24.8_22657.20170607
[ 1147.655602] RTL871X: HW EFUSE
[ 1147.658631] RTL871X: hal_com_config_channel_plan chplan:0x20
[ 1147.991298] RTL871X: rtw_regsty_chk_target_tx_power_valid return _FALSE for band:0, path:0, rs:0, t:-1
[ 1148.010426] RTL871X: rtw_ndev_init(wlan0) if1 mac_addr=68:b9:d3:ca:4c:4c
[ 1148.041449] RTL871X: module init ret=0
[ 1150.034576] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
root@ATK-IMX6U:~/shell/wifi# [ 1151.555831] RTL871X: set bssid:00:00:00:00:00:00

root@ATK-IMX6U:~/shell/wifi#
```

图 3.24 4 执行 SDIO WIF 初始化脚本

扫描附近无线网络信息并打印，使用 iwlist 扫描到本公司的无线网络名称“ALIENTEK-YF”带有中文的热点名称扫描不出或者乱码看不到，测试时确保热点名是英文字符。

```
ifconfig wlan0 up                    // 开启 SDIO WIFI
iwlist wlan0 scan | grep SSID
```

```
root@ATK-IMX6U:~/driver/rtl8189# iwlist wlan0 scan | grep SSID
                    ESSID:"ALIENTEK-FAE"
                    ESSID:"Wireless_BY"
                    ESSID:""
                    ESSID:"@PHICOMM_94"
                    ESSID:""
                    ESSID:""
                    ESSID:"hqfs2.4g"
                    ESSID:"ChinaNet-PTHm"
                    ESSID:"hqtz2.4g"
                    ESSID:"ALIENTEK-YF"
                    ESSID:"1004"
                    ESSID:""
                    ESSID:"ZZK"
                    ESSID:""
                    ESSID:"ChinaNet-JqHm"
                    ESSID:""
                    ESSID:"1DW2.4g"
                    ESSID:"ALIENTEK-HYS"
                    ESSID:"Admin2.4G"
                    ESSID:"TP-LINK_BF32"
                    ESSID:"DW2.4"
                    ESSID:"ChinaNet-Hr7s"
                    ESSID:"ousu"
root@ATK-IMX6U:~/driver/rtl8189#
```

扫描到本公司热点名称

图 3.24 5 扫描附近无线网络信息

将下面的内容在当前/home/root/目录下编辑成一个 wpa_supplicant.conf 配置文件。

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

热点名有密码时，配置/etc/wpa_supplicant.conf 文件内容如下

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1
```

```
network={
        ssid="ALIENTEK-YF"
        psk="1590202****"
}
```

热点名无密码时，配置/etc/wpa_supplicant.conf 文件内容如下

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1
network={
        ssid="ALIENTEK-YF  "
        key_mgmt=NONE
}
```

解释：

- ssid 为无线网络名称
- psk 为无线网络密码

使用下面的指令来连接配置的无线网络，当看到“wlan0: CTRL-EVENT-CONNECTED - Connection to ec:88:8f:77:d6:da completed [id=0 id_str=]”这句话表明连接成功。如果连接不成功会提示“wlan0: WPA: 4-Way Handshake failed - pre-shared key may be incorrect”这样的话，请用户检查无线网络信息是否有误。

```
wpa_supplicant -Dnl80211 -c /etc/wpa_supplicant.conf -i wlan0 &
```

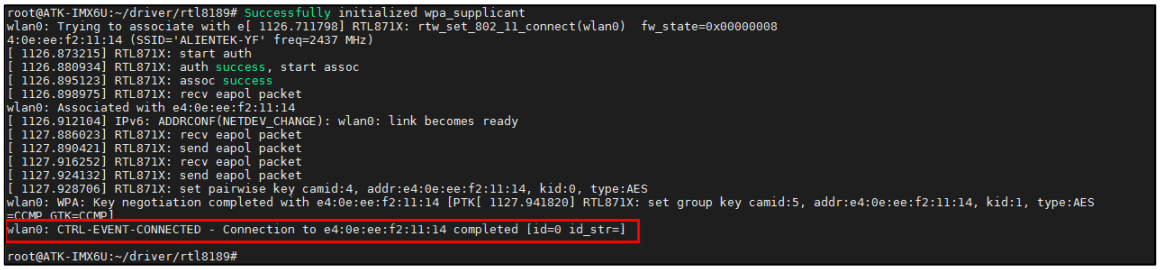


图 3.24 6 连接无线网络热点

连接成功后，为 wlan0 获取 ip

```
udhcpc -i wlan0
```

图 3.24 7 为 wlan0 获取 ip

执行下面的指令测试上网，参数“-I”指定网卡。这里要指定 wlan0 网卡。可按“Ctrl + c”终止 ping 指令。出现如下结果，说明 ping 百度成功。

```
route add default gw 192.168.101.1 dev wlan0    // 设置默认网关为 SDIO WIFI 的网关
ping www.baidu.com -I wlan0
```

```
root@ATK-IMX6U:~/driver/rtl8189# ping www.baidu.com -I wlan0
PING www.a.shifen.com (14.215.177.39) from 192.168.101.177 wlan0: 56(84) bytes of data.
64 bytes from 14.215.177.39: icmp_seq=1 ttl=54 time=187 ms
64 bytes from 14.215.177.39: icmp_seq=2 ttl=54 time=8.95 ms
64 bytes from 14.215.177.39: icmp_seq=3 ttl=54 time=20.6 ms
64 bytes from 14.215.177.39: icmp_seq=4 ttl=54 time=20.5 ms
64 bytes from 14.215.177.39: icmp_seq=5 ttl=54 time=114 ms
^C
--- www.a.shifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 8.952/70.270/187.031/69.657 ms
root@ATK-IMX6U:~/driver/rtl8189#
```

图 3.24 8 ping 百度测试联网功能

3.24.1 Staion(上网) 模式

为了方便用户测试，我们编写了一个 alientek_sdio_wifi_setup.sh 脚本，路径在/home/root/shell/wifi/下。此脚本是后面新版本文件系统添加的，请更新文件系统到 v2.3 及以上。

与 3.22.1 小节操作一样。请参考 3.22.1 的操作步骤，再执行下面的指令。

进入/home/root/shell/wifi/目录下。

```
cd /home/root/shell/wifi
```

请确保 SDIO WIFI 已经初始化。否则需要执行 SDIO WIFI 初始化脚本。

```
/home/root/alientek_sdio_wifi_init.sh
```

执行下面的脚本，输入个人的无线网络名称，及密码。

```
source ./alientek_sdio_wifi_setup.sh -m station -i ALIENTEK-YF  -p 1590202**** -d wlan0
```

3.2.4.2 SoftAP(热点) 模式

与 3.22.2 小节操作一样。请参考 3.22.2 的操作步骤，再执行下面的指令。

```
source ./alientek_sdio_wifi_setup.sh -m softap -d wlan0
```

3.24.3 Bridge(桥接)模式

与 3.22.3 小节操作一样。请参考 3.22.3 的操作步骤，再执行下面的指令。

```
source ./alientek_usb_wifi_setup.sh -m bridge -d wlan0 -e eth0
```

3.25 RGB 转 HDMI 模块测试

准备正点原子 RGB-HDMI V1.3 模块，模块需要另外购买，比较便宜，如下图，LCD 接口在背面。搭配一段 5cm 左右的 fpc 软排线。请不要使用长于 5cm 的 fpc 软排线，防止因为 fpc 软排线太长，信号受到干扰。RGB 转 HDMI 需要使用 RGB 转 HDMI 的驱动，HDMI 线连接显示器端支持热插拔。所以在上电前需要连接好模块与支持 HDMI 的显示器。RGB 转 HDMI 模块最大支持 1366*768 60Hz 输出。因为 IMX6U 只支持 1366*768 60Hz 的 LCD 输出。分辨率再高时钟就会波形失真，不能高于这个分辨率。考虑到有些显示器不支持 1366*768 60Hz 的 HDMI 输入，所以我们出厂就配置了 1280*720 60Hz，标准的 720p 的 HDMI 输出。

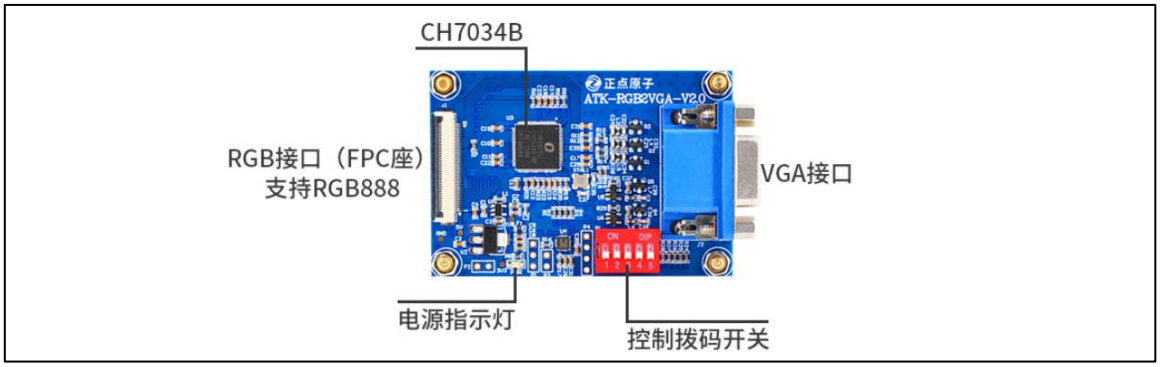
3.25 1 正点原子 RGB 转 HDMI 模块

和正点原子所有 RGB 屏一样，RGB 转 HDMI 模块可以被出厂的系统识别，无需更换设备树。

先将显示器上电，用 HMDI 连接线连接模块与显示器。用 5cm 的 fpc 软排线接到模块上，接到开发板 RGBLCD 接口处。开发板上电启动，HDMI 就可以显示出图像了。（**注意事项:HDMI 接口处一定要插到底**）.

3.26 RGB 转 VGA 模块测试

准备 ATK-RGB2VGA-V2.0 模块（此模块最大支持 1920*1080 60Hz 输出），模块需要另外购买。如下图，搭配一段 5cm 左右的 fpc 软排线。请不要使用长于 5cm 的 fpc 软排线，防止因为 fpc 软排线太长，信号受到干扰。相对于 RGB 转 HDMI 模块来说，RGB 转 VGA 模块更加稳定。因为它不需要 Linux 驱动，支持热插拔。RGB 信号直接输入到这个模块上就可以自动转换成 VGA 信号输出。供有需要 VGA 信号输出的用户使用。



3.26 1 正点原子 RGB 转 VGA 模块

和正点原子所有 RGB 屏一样，RGB 转 VGA 模块可以被出厂的系统识别，无需更换设备树。因为 IMX6U 最大支持 1366*768 60Hz 的 LCD 输出。分辨率再高时钟就会波形失真，不能高于这个分辨率。ALPHA 开发板 RGB 转 VGA 模块设备树默认配置了 1366*768 60Hz 的输出。将 RGB 转 VGA 模块拨码至 10000，其他分辨率需要看 ATK-RGB2VGA 模块用户手册，请在淘宝买页处下载 RGB 转 VGA 模块用户手册资料。开发板上电，开发板自动识别 VGA 模块，VGA 信号自动输出。

3.27 关机、重启和休眠

由于 I.MX6ULL 芯片使用挂起和关机指令都会对芯片内部寄存器写了某个值，而板子 RTC 电池与板子构成一个回路，所以如果需要输入挂起或者关机指令的用户，不能装 RTC 电池。否

则需要长按 ON_OFF 按钮或者短接地 ON_OFF 管脚复位芯片内部寄存器的值才能重新开机。（注：2021 年 6 月 18 日更新出厂系统后，可以使用关机指令或挂起指令来关机。）

```
init 0      // 挂起
init 6      // 重启
halt        // 挂起
poweroff // 关机
reboot      // 重启
shutdown // 挂起
```

常用的休眠方式有 freeze，standby， mem， disk 但是默认出厂系统不支持 disk 方式。

- freeze: 冻结 I/O 设备,将它们置于低功耗状态,使处理器进入空闲状态，唤醒最快，耗电比其它 standby， mem，disk 方式高
- standby:除了冻结 I/O 设备外，还会暂停系统，唤醒较快，耗电比其它 mem， disk 方式高
- mem:将运行状态数据存到内存，并关闭外设，进入等待模式,唤醒较慢，耗电比 disk 方式高
- disk: 将运行状态数据存到硬盘，然后关机，唤醒最慢

输入下面指令，使用 mem 方式将系统休眠，唤醒时换 KEY0 按键。

```
killall  apmd                                    // 先杀掉进阶电源管理服务程序
echo mem > /sys/power/state
```

```
root@ATK-IMX6U:~# echo mem > /sys/power/state
[74554.203295] PM: Syncing filesystems ... done.
[74555.257338] Freezing user space processes ... (elapsed 0.001 seconds) done.
[74555.266779] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.
[74555.275388] Suspending console(s) (use no_console_suspend to debug)
```

图 3.27 1 开启休眠模式

3.28 查看 CPU 主频

目前正点原子在售的 I.MX6U 都是 CPU 主频 800MHz（实 792MHz）的。我们可以使用如下指令查看出厂系统 CPU 主频。

查看可用主频，使用如下指令，可以看到有 198MHz、396MHz、528MHz 和 792MHz 的主频可用。

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
```

```
root@ATK-IMX6U:~# cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
198000 396000 528000 792000
root@ATK-IMX6U:~#
```

图 3.28 1 查看可用主频

输入 cpufreq-info 查看 cpu 运行的频率，由于出厂内核配置了 CPU 主频为“Performance”（此模式不考虑耗电，最高性能），直接运行在最高主频上。所以看到 792MHz 处是 100%。

```
cpufreq-info
```

```
root@ATK-IMX6U:~# 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: imx6q-cpufreq
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency: 91.0 us.
  hardware limits: 198 MHz - 792 MHz
  available frequency steps: 198 MHz, 396 MHz, 528 MHz, 792 MHz
  available cpufreq governors: interactive, conservative, userspace, powersave, performance
  current policy: frequency should be within 198 MHz and 792 MHz.
                  The governor "performance" may decide which speed to use
                  within this range.
  current CPU frequency is 792 MHz (asserted by call to hardware).
  cpufreq stats: 198 MHz:0.00%, 396 MHz:0.00%, 528 MHz:0.00%, 792 MHz:100.00%  (1)
root@ATK-IMX6U:~# _
```

图 3.28 2 查看 CPU 主频相关信息

那么是否可以无需重新编译 Linux 源码来修改 CPU 的主频呢，答案是可以的！

输入下面的指令，申请用户空间控制 CPU 主频。

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

输入下面指令将 CPU 主频修改为 198MHz，注意不是任意主频都支持，必须是上面支持的那几种频率！

```
echo 198000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

再查看当前的 CPU 主频，修改成了低主频，明显的是 Qt 桌面滑动变卡了，说明此主频生效！

```
cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq
```

```
root@ATK-IMX6U:~# echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
root@ATK-IMX6U:~# echo 198000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
root@ATK-IMX6U:~# cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq
198000
root@ATK-IMX6U:~#
```

图 3.28 3 查看当前 CPU 主频

3.29 USB 蓝牙测试

准备一个 USB 蓝牙，请自行在网上购买（十几二十块 RMB 就可以买到一个），如下图。使用的是 CSR 蓝牙芯片，一般免驱动。可在 Window/Linux 下直接使用。下图的测试的是蓝牙 4.0 版本的蓝牙模块（蓝牙 5.0 请自行测试），此种模块默认不需要输入 Pin 码配对。

图 3.29 1 USB 蓝牙模块

在正点原子的出厂文件系统里有 BlueZ，一种开源的蓝牙协议栈，并带有相关的蓝牙测试工具，可以很方便的测试蓝牙。

蓝牙支持热插拔，所以在开发板开机后插上 USB 蓝牙模块如下图。

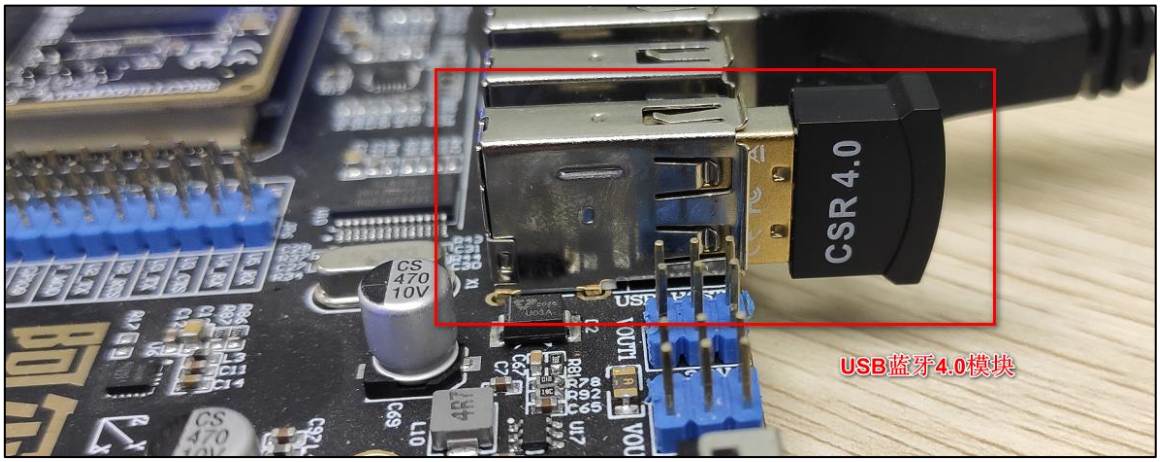


图 3.29 2 ALPHA 开发板插上 USB 蓝牙模块

开启蓝牙，编辑一个脚本。v2.3 版本文件系统里/home/root/shell/bluetooth/已经提供了 bluetooth_start.sh 脚本，可以直接执行，如果您的文件系统不是 v2.3 版本，需要用户自己制作 bluetooth_start.sh 脚本。

```
vi bluetooth_start.sh
```

在 blue_start.sh 添加以下内容，按 Esc 键后，再按:wq 保存退出。注意有个“:”号不要漏了。

```
#!/bin/sh
#由于出厂系统已经使用 rfkill 管射频之类的设备，在没有用到这类设备时是关闭它的射频的，
#达到省电的目的，现在我们解锁射频
rfkill unblock all
sleep 1
#开启蓝牙网络（使能 USB 蓝牙设备）
hciconfig hci0 up
sleep 1
#启动 bluetoothd 服务
/etc/init.d/bluetooth start
sleep 2
#开启蓝牙被扫描
hciconfig hci0 piscan
```

赋予 bluetooth_start.sh 脚本可执行权限

```
chmod +x bluetooth_start.sh
```

启动蓝牙。

```
./bluetooth_start.sh
```

```
root@ATK-IMX6U:~# vi bluetooth_start.sh
root@ATK-IMX6U:~# chmod +x bluetooth_start.sh
root@ATK-IMX6U:~# ./bluetooth_start.sh
Starting bluetooth
bluetoothd
root@ATK-IMX6U:~#
```

图 3.29 3 启动蓝牙相关服务

在上面启动了蓝牙之后，用手机可扫描到蓝牙名称 BlueZ 5.37。如下图。

图 3.29 4 手机识别到的蓝牙名称

之所以会扫描出 BlueZ 5.37 这个名称，是系统里 BlueZ 的版本为 5.37。我们可以修改这个蓝牙的名称。

使用 hciconfig 指令，关于 hciconifg 指令的用法，可输入“hciconfig -h”自行查看，这里不多解释，尽量把常用的写出来。

查看当前蓝牙的名称。

```
hciconfig hci0 name
```

```
root@ATK-IMX6U:~# hciconfig hci0 name
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:DA:71:10  ACL MTU: 310:4  SCO MTU: 64:8
        Name: 'BlueZ 5.37'
root@ATK-IMX6U:~#
```

图 3.29 5 查看当前蓝牙名称

输入下面的指令修改蓝牙的名称，再查看。（注意：可能有些蓝牙模块可能修改不能成功，手机扫描出的名称还是 BlueZ 5.37）

```
hciconfig hci0 name imx6ull
hciconfig hci0 name
```

```
root@ATK-IMX6U:~# hciconfig hci0 name imx6ull
root@ATK-IMX6U:~# hciconfig hci0 name
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1A:7D:DA:71:10  ACL MTU: 310:4  SCO MTU: 64:8
        Name: 'imx6ull'
root@ATK-IMX6U:~#
```

图 3.29 6 修改蓝牙名称

3.29.1 蓝牙建立连接

我们可以使用 bluetoothctl 蓝牙工具进行与蓝牙进行交互，输入指令后可以看到蓝牙的 MAC 地址。

```
bluetoothctl
```

```
root@ATK-IMX6U:~# bluetoothctl
[NEW] Controller 00:1A:7D:DA:71:10 imx6ull [default]
[bluetooth]#
```

图 3.29.1 1 进入 bluetoothctl 交互

输入 help 查看 bluetoothctl 的用法。

```
root@ATK-IMX6U:~# bluetoothctl
[NEW] Controller 00:1A:7D:DA:71:10 imx6ull [default]
[bluetooth]# help
Available commands:
  list                         List available controllers
  show [ctrl]                  Controller information
  select <ctrl>                Select default controller
  devices                      List available devices
  paired-devices               List paired devices
  power <on/off>               Set controller power
  pairable <on/off>            Set controller pairable mode
  discoverable <on/off>        Set controller discoverable mode
  agent <on/off/capability>    Enable/disable agent with given capability
  default-agent                Set agent as the default one
  set-scan-filter-uuids [uuid1 uuid2 ...] Set scan filter uuids
  set-scan-filter-rssi [rssi] Set scan filter rssi, and clears pathloss
  set-scan-filter-pathloss [pathloss] Set scan filter pathloss, and clears rssi
  set-scan-filter-transport [transport] Set scan filter transport
  set-scan-filter-clear        Clears discovery filter.
  scan <on/off>                Scan for devices
  info [dev]                   Device information
  pair [dev]                   Pair with device
  trust [dev]                  Trust device
  untrust [dev]                Untrust device
  block [dev]                  Block device
  unblock [dev]                Unblock device
  remove <dev>                 Remove device
  connect <dev>                Connect device
  disconnect [dev]             Disconnect device
  list-attributes [dev]        List attributes
  select-attribute <attribute> Select attribute
  attribute-info [attribute] Select attribute
  read                         Read attribute value
  write <data=[xx xx ...]>     Write attribute value
  notify <on/off>              Notify attribute value
  register-profile <UUID ...> Register profile to connect
  unregister-profile           Unregister profile
  version                      Display version
  quit                         Quit program
[bluetooth]#
```

图 3.29.1 2 查看 bluetoothctl 相关用法

继续输入下面的指令，开启蓝牙的电源、代理。

```
power on
agent on
default-agent
```

```
[bluetooth]# power on
[bluetooth]# agent on
[bluetooth]# default-agent
No agent is registered
Changing power on succeeded
Agent registered
[bluetooth]#
```

图 3.29.1 3 开启相关指令

开启蓝牙扫描。输入 scan on 等待扫描之后结果，输入 scan off 关闭扫描。主要得到我们的测试手机的蓝牙 MAC 地址与名称。

```
scan on
scan off
```

```
[bluetooth]# scan on
Discovery started
[CHG] Controller 00:1A:7D:DA:71:10 Discovering: yes
[NEW] Device 6B:8A:9C:75:DE:B7 6B-8A-9C-75-DE-B7
[NEW] Device 9C:F6:DD:30:EC:A4 9C-F6-DD-30-EC-A4
[NEW] Device 6B:00:00:DD:BE:96 6B-00-00-DD-BE-96
[NEW] Device 5B:25:56:02:DA:88 5B-25-56-02-DA-88
[NEW] Device 6D:B4:6D:24:61:D1 6D-B4-6D-24-61-D1
[NEW] Device E0:1F:88:34:15:3F mi_10ult
[NEW] Device AC:C1:EE:12:06:E3 小米手机
[NEW] Device 60:AB:67:65:93:E8 alientek_test
[NEW] Device 90:F0:52:90:E9:19 bin
[NEW] Device 78:02:F8:08:AC:DB 小米手机000
[bluetooth]# scan off
[CHG] Device 78:02:F8:08:AC:DB RSSI is nil
[CHG] Device 90:F0:52:90:E9:19 RSSI is nil
[CHG] Device 60:AB:67:65:93:E8 RSSI is nil
[CHG] Device AC:C1:EE:12:06:E3 RSSI is nil
[CHG] Device E0:1F:88:34:15:3F RSSI is nil
Discovery stopped
[CHG] Controller 00:1A:7D:DA:71:10 Discovering: no
[bluetooth]#
```


图 3.29.1 4 扫描测试手机的蓝牙

主动连接我们的测试手机蓝牙。输入 connect 指令。

指令格式为 connect XX:XX:XX:XX:XX:XX，XX:XX:XX:XX:XX:XX 请填上个人手机蓝牙的 MAC 地址。

```
trust 60:AB:67:65:93:E8                    // 连接前最好信任这个设备
connect 60:AB:67:65:93:E8
```

```
[bluetooth]# connect 60:AB:67:65:93:E8
Attempting to connect to 60:AB:67:65:93:E8
[CHG] Device 60:AB:67:65:93:E8 Connected: yes
[CHG] Device 60:AB:67:65:93:E8 Modalias: bluetooth:v038Fp1200d1436
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001103-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001105-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 0000110a-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 0000110c-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001112-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001115-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001116-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 0000111f-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 0000112f-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001132-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001200-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001800-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 00001801-0000-1000-8000-00805f9b34fb
[CHG] Device 60:AB:67:65:93:E8 UUIDs: 98b97136-36a2-11ea-8467-484d7e99a198
Request confirmation
[agent] Confirm passkey 445910 (yes/no): yes
[CHG] Device 60:AB:67:65:93:E8 Paired: yes
Connection successful
[alientek_test]#
```

图 3.29.1 5 连接手机蓝牙

默认确认 PIN 码，或者不需要输入 PIN 码，根据蓝牙的不同类型而不同。选择购买的蓝牙最好是不需要输入 PIN 码，方便连接。

当我们的代理开启时，蓝牙自动发送随机 PIN 码到手机（有些蓝牙模块可能需要手动输入 PIN 码），我们点击配对即可连接。

图 3.29.1 6 与手机配对

连接成功后，手机上也显示已经连接。如下图。

图 3.29.1 7 与手机连接成功

3.29.2 蓝牙音乐

ALPHA	MINI
本实验支持	不支持，无音频。可以用 USB 声卡尝试做本实验

查看文件系统版本，请在 1.2.2 小节查看**文件系统历史版本**，文件系统需要在 V2.0 以上版本才支持蓝牙音乐。

如下图，在手机与板子上的 USB 蓝牙模块连接后，手机会识别蓝牙成一个耳机设备，SBC 是一种音频格式，音频一般，大多数蓝牙耳机都是这种格式。

图 3.29.2 1 手机显示蓝牙为 SBC 耳机类型设备

此时手机打开音乐，ALPHA 的底板喇叭就会播放手机上的歌曲。

3.29.3 蓝牙传送文件

在 bluetoolctl 交互终端输入 discoverable on，配置蓝牙可被发现。如果没有在 bluetoolctl 开启可被发现，手机向板子上的蓝牙发送文件时，可能会找不到蓝牙设备。

```
[bluetooth]# discoverable on
Changing discoverable on succeeded
[bluetooth]#
```

图 3.29.3 1 开启蓝牙被发现

退出 bluetoolctl，输入 exit 或者 quit 退出。因为我们在 3.29.1 小节已经了配对，我们可以直接连接。

在终端输入如下指令，先开启 dbus 服务，启动 obexctl，配置蓝牙发送与接收，使用的是 OBEX 协议协输。注意“`”这个符号，是键盘左上角数字 1 隔壁的字符“`”。开启成功和连接成功如下。

```
eval `dbus-launch --sh-syntax`
/usr/libexec/bluetooth/obexd -r /home/root -a -d & obexctl        // 进入 obexctl 交互模式
connect 60:AB:67:65:93:E8                  // 连接手机蓝牙，请填写个人手机蓝牙 MAC
```

obexd 参数解释：

（1） -r：设置用于接收的路径

（2） -a：设置自动接收

（3） -d：打印 debug 信息

开启命令行测试工具 obexctl

```
root@ATK-IMX6U:~# eval `dbus-launch --sh-syntax`
root@ATK-IMX6U:~# /usr/libexec/bluetooth/obexd -r /home/root -a -d & obexctl
[1] 876
[NEW] Client /org/bluez/obex
[obex]# connect 60:AB:67:65:93:E8
Attempting to connect to 60:AB:67:65:93:E8
[NEW] Session /org/bluez/obex/client/session0 [default]
[NEW] ObjectPush /org/bluez/obex/client/session0
Connection successful
[60:AB:67:65:93:E8]#
```

图 3.29.3 2 进入 obexctl 交互模式

发送/home/root/blue_start.sh 这个脚本到手机，输入下面的指令。由于脚本文件较小，传输比较快。

```
send /home/root/blue_start.sh
```

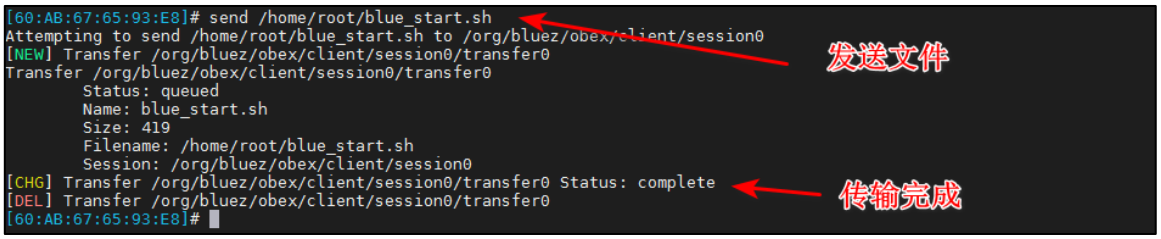


图 3.29.3 3 发送文件

手机接收到发来的文件弹窗，我们点击接受即可。

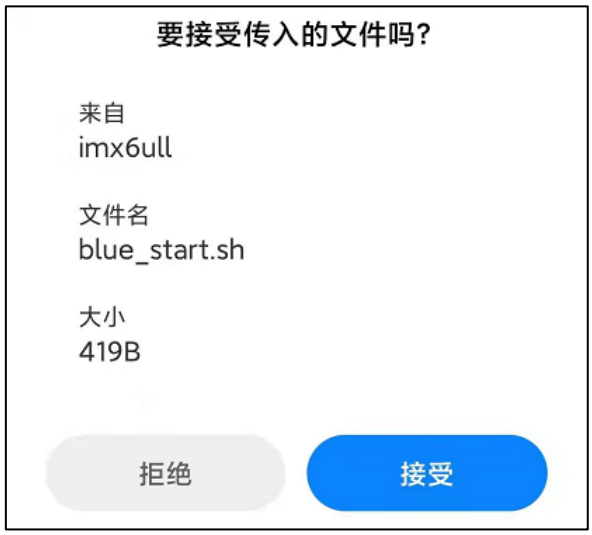


图 3.29.3 4 手机收到板子的 USB 蓝牙发送的文件

反过来也一样，手机也可以向板子发送文件，默认存放在/home/root 目录下。

3.29.4 更多蓝牙工具

使用 l2ping 可以像 ping 命令一样检查蓝牙是否能在线

```
l2ping  蓝牙地址
```

查看蓝牙设备的可用服务

```
sdptool browse 蓝牙地址
```

3.30 DS18B20 测试

开发板用于测试 DS18B20 对应的管脚关系如下：

开发板	GPIO1
ALPHA/Mini	DATA

DS18B20 简介：

DS18B20 是由 DALLAS 半导体公司推出的一种的“一线总线”接口的温度传感器。与传统的热敏电阻等测温元件相比，它是一种新型的体积小、适用电压宽、与微处理器接口简单的数字化温度传感器。一线总线结构具有简洁且经济的特点，可使用户轻松地组建传感器网络，从而为测量系统的构建引入全新概念，测量温度范围为-55~+125℃ ，精度为±0.5℃。3-5.5 V 的电压范围，采用多种封装形式，从而使系统设计灵活、方便等。

此实验需要准备 DS18B20 模块，可在正点原子淘宝店购买。

图 3.30 1 DS18B20 模块

DS18B20 模块的管脚排列如下图， 以**正视图为准**。

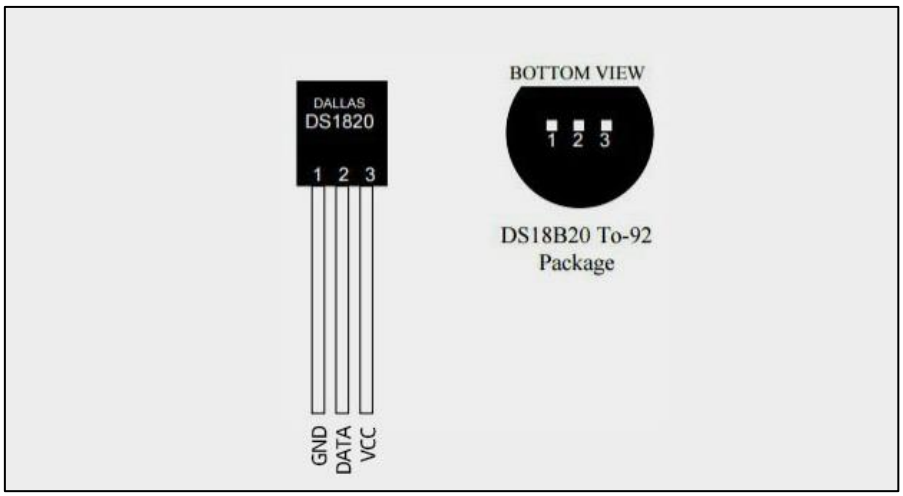


图 3.30 2 DS18B20 正视图与俯视图

由于开发板管脚资源有限，使用紧张。没有单独留出一个座子给 DS18B20 和 DHT11 使用。查看原理图可知，GPIO1 管理上 ATK MODULE 座子上。所以我们可以通过杜绑线来连接 DS18B20 到 ATK MODULE 座子上的 GPIO1 管脚上。

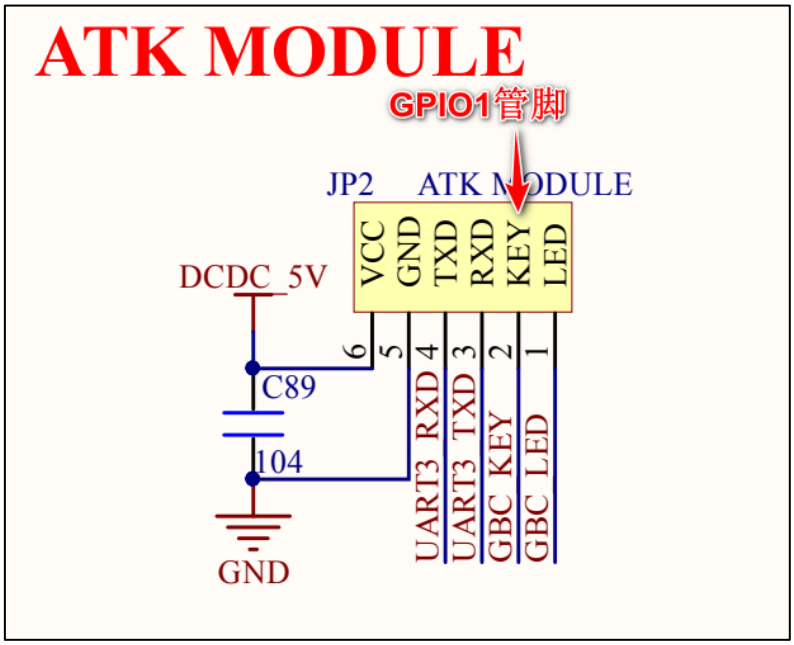


图 3.30 3 底板 ATK 模块原理图

那么我们就可以使用三根杜邦线（一头公，另一头母的），连接 DS18B20 模块。如下图。

图 3.30 4 DS18B20 与杜邦线连接

然后根据上文的 DS18B20 的正视图，有字的一面为正视图。将 DATA 脚接到 ATK MODULE 座子上丝印为 KEY 的座孔上。然后再将 VCC，与 GND 都接在 ATK MODULE 的孔位上。（注意不要将 DS18B20 传感器的 VCC 与 GND 接反，否则会将 DS18B20 烧坏，若接反，模块会讯速发热，请快速断开！检查接法）。如下图，正确的接法如下。下图为 APLHA 底板与 DS18B20 接法，Mini 底板同理。

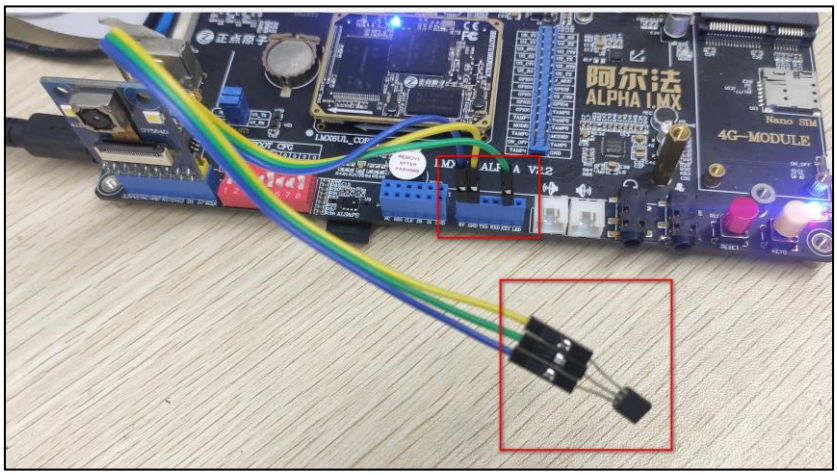


图 3.30 5 DS18B20 与底板连接

接好模块后，我们需要加载驱动。正点原子为 DS18B20 编写了驱动模块。驱动在文件系统路径/home/root/driver/ds18b20/ds18b20.ko。所以我们直接加载这个驱动即可。（注：此驱动需要更新到 v2.2 版本的文件系统才有）。

```
insmod /home/root/driver/ds18b20/ds18b20.ko
```

```
root@ATK-IMX6U:~# insmod /home/root/driver/ds18b20/ds18b20.ko
[ 9061.945934] ds18b20 ds18b20: ds18b20 device and driver matched successfully!
root@ATK-IMX6U:~#
```

图 3.30 6 加载 DS18B20 驱动

读取 DS18B20 的温度传感器的值。

```
cat /sys/class/misc/ds18b20/value
```

```
root@ATK-IMX6U:~# cat /sys/class/misc/ds18b20/value
293750
root@ATK-IMX6U:~#
```

图 3.30 7 读取 DS18B20 的数据

得出来的数值除以 10000，就是实际的温度数据，如上，得到的环境温度为 29.375 度（注：精度为±0.5℃）。如果不能获取数据，请检查是传感器是否连接稳定，及接法是否正确。

3.31 DHT11 测试

开发板用于测试 DHT11 对应的管脚关系如下：

开发板	GPIO1
ALPHA/Mini	Dout

DHT11 简介：

DHT11 是一款湿温度一体化的数字传感器。该传感器包括一个电阻式测湿元件和一个 NTC 测温元件，并与一个高性能 8 位单片机相连接。通过单片机等微处理器简单的电路连接就能够实时的采集本地湿度和温度。DHT11 与单片机之间能采用简单的单总线进行通信，仅仅需要一个 I/O 口。传感器内部湿度和温度数据 40Bit 的数据一次性传给单片机，数据采用校验和方式进行校验，有效的保证数据传输的准确性。DHT11 功耗很低，5V 电源电压下，工作平均最大电流 0.5mA。

此实验需要准备 DHT11 模块，可在正点原子淘宝店购买。

图 3.31 1 DHT11 模块

正点原子淘宝 DHT11 的技术参数如下：

工作电压范围：3.3V-5.5V

工作电流： 平均 0.5mA

输出：单总线数字信号

测量范围：湿度 20~90％RH，温度 0~50℃

精度 ：湿度±5%，温度±2℃

分辨率 ：湿度 1%，温度 1℃

DHT11 模块的管脚排列如下图，下图为正面视图，有孔的一面为正面。

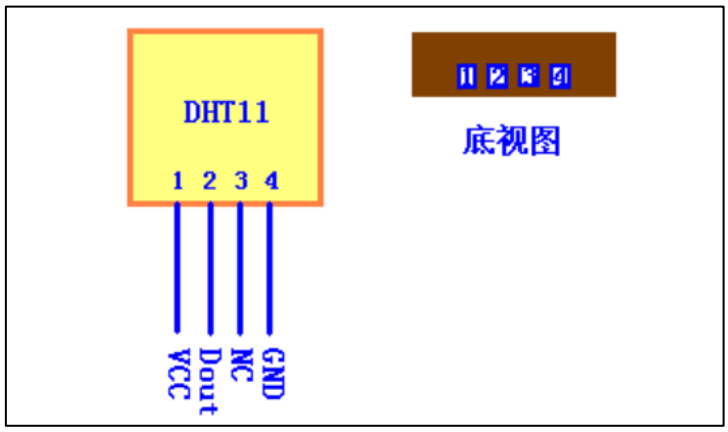


图 3.31 2 DHT11 正视图与底视图

由于开发板管脚资源有限，使用紧张。没有单独留出一个座子给 DS18B20 和 DHT11 使用。查看原理图可知，GPIO1 管理上 ATK MODULE 座子上。所以我们可以通过杜绑线来连接 DHT11 到 ATK MODULE 座子上的 GPIO1 管脚上。

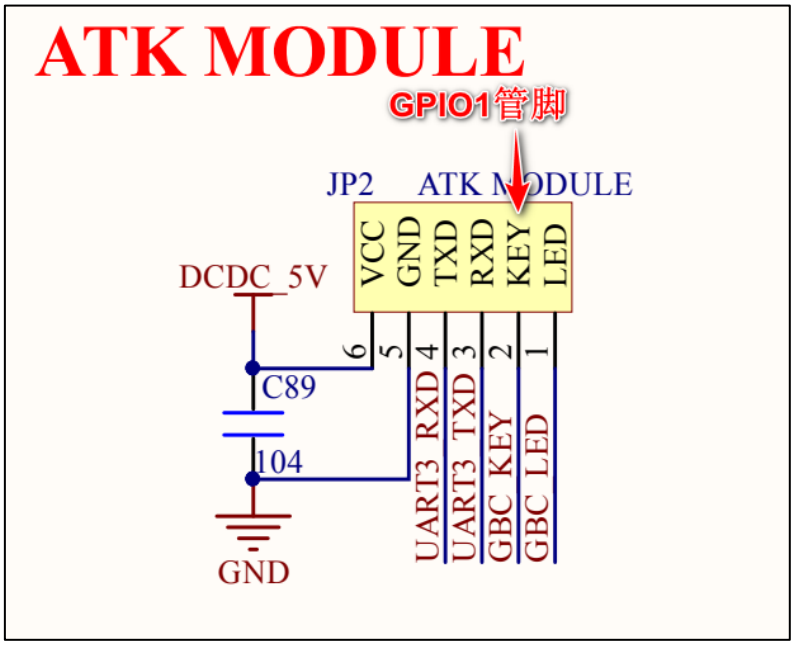


图 3.31 3 ATK 模块原理图

那么我们就可以使用三根杜邦线（一头公，另一头母的），其中 NC 脚不用接，连接 DHT11 模块。如下图。

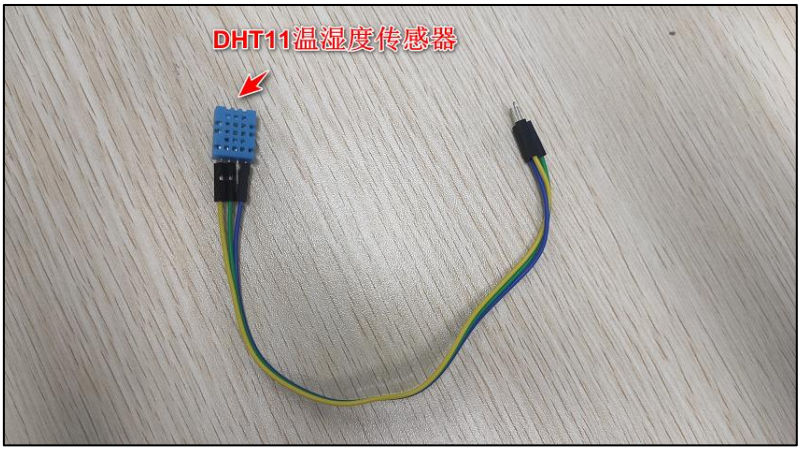


图 3.31 4 DHT11 与杜邦线连接

然后根据上文的 DHT11 的正视图，有孔的一面为正视图。将 DATA 脚接到 ATK MODULE 座子上丝印为 KEY 的座孔上。然后再将 VCC，与 GND 都接在 ATK MODULE 的孔位上。如下图，正确的接法如下。注意，有孔的一面应朝上，不应被其他异物遮挡。下图为 APLHA 底板与 DHT11 接法，Mini 底板同理。

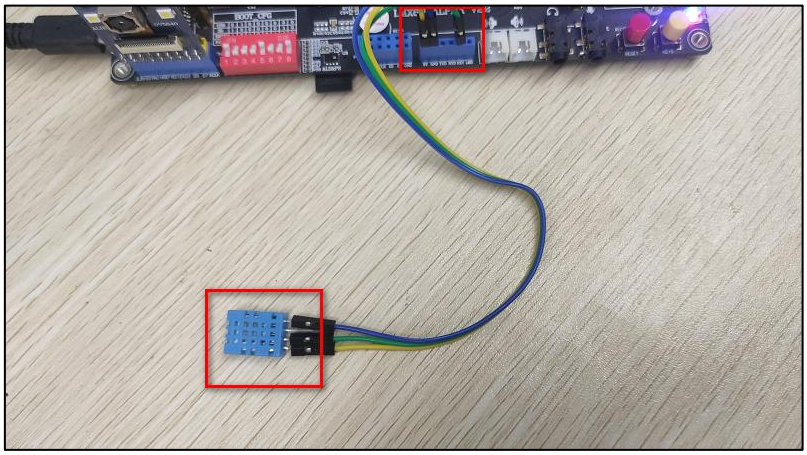


图 3.31 5 DHT11 与底板连接

如果您在上一小节，测试过 DS18B20，已经加载过 DS18B20 驱动的话，我们应该卸载 DS18B20 的驱动，因为它们是共用同一个管脚的。不能同时加载。执行下面的指令卸载 DS18B20 驱动。

```
rmmod /home/root/driver/ds18b20/ds18b20.ko
```

```
root@ATK-IMX6U:~# rmmod /home/root/driver/ds18b20/ds18b20.ko
[21242.323080] ds18b20 ds18b20: DS18B20 driver has been removed!
root@ATK-IMX6U:~# _
```

图 3.31 6 卸载 DS18B20 驱动

接好模块后，我们需要加载驱动。正点原子为 DHT11 编写了驱动模块。驱动在文件系统路径/home/root/driver/dht11/ dht11.ko。所以我们直接加载这个驱动即可。（注：此驱动需要更新到 v2.2 版本的文件系统才有）。

```
insmod /home/root/driver/dht11/dht11.ko
```

```
root@ATK-IMX6U:~# insmod /home/root/driver/dht11/dht11.ko
[21679.325162] dht11 dht11: dht11 device and driver matched successfully!
root@ATK-IMX6U:~#
```

图 3.31 7 加载 DHT11 驱动

读取 DHT11 的湿度与温度传感器的值。

```
cat /sys/class/misc/dht11/value
```

```
root@ATK-IMX6U:~# cat /sys/class/misc/dht11/value
10.0,30.0
root@ATK-IMX6U:~#
```

图 3.31 8 读取 DHT11 数据

得出来的数值前面的数据为湿度，后面的数据为温度。注意精度湿度±5%，温度±2℃。数据有小数是为了数据好看。如果不能获取数据，请检查是传感器是否连接稳定，及接法是否正确。

第四章 ATK I.MX6U 交叉编译

本章主要简介正点原子 I.MX6U 交叉编译工具的使用。

章节前言：

搭建交叉编译环境是学习 Linux 很重要的一步，正点原子提供两种交叉编译工具链。这两种交叉编译工具链解释如下图。这两种编译器都有各自的作用，我们只要在相应的实验里使用指定的交叉编译工具链即可，切勿混着用。同时它们的使用方法是不一样的，不能生搬硬套！第一种交叉编译器使用的方法，请参考本文档的 4.2~4.7 小节。第二种交叉编译器请参考【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南。

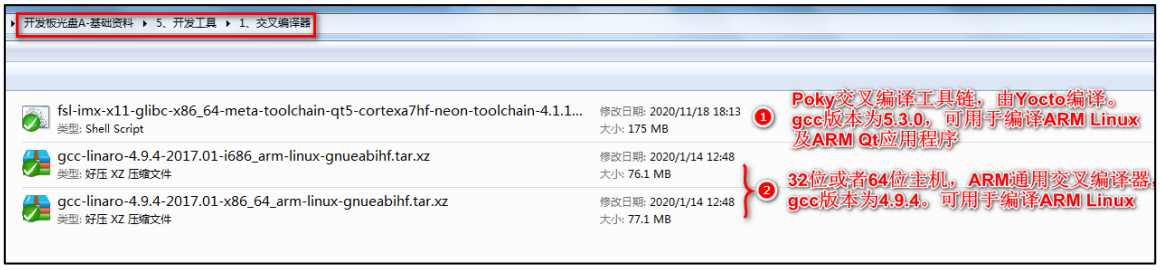


图 4 1 资料提供的两种交叉编译工具链

很多用户容易将上面两种交叉编译工具搞混。我们只需要知道上面的第二种通用的交叉编译器去学习【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南这本教程。第一种 Poky 编译器是 Yocto 项目编译出的，常用于编译开发板光盘 A-基础资料->1、例程源码->3、正点原子 Uboot 和 Linux 出厂源码及快速编译 Qt 应用程序到开发板上运行。

4.1 安装通用 ARM 交叉编译工具链

详细请看【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南 V1.x.pdf 的第 4.3 小节，这里不再重复写了。学习教程时需要再看，我们继续往下看其他部分。

4.2 安装 Poky 交叉编译工具链

把开发板光盘 A-基础资料->5、开发工具->1、交叉编译器->fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh 拷贝到 Ubuntu 虚拟机
如下图本文已经把交叉编译工具拷贝到了 Ubuntu 虚拟机。

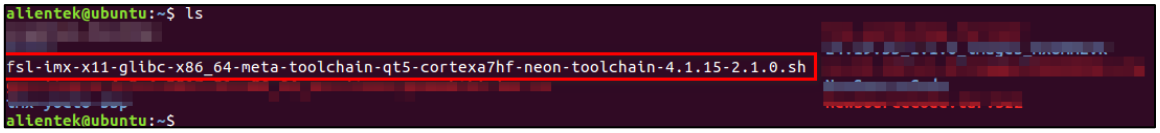


图 4.2.1 拷贝 sdk 工具包到 ubuntu 系统里

执行下面的指令修改脚本的权限，修改权限后可以看到此脚本颜色显示改变，说明修改成功。

```
chmod u+x fsl-imx-x11-glibc-x86_64-meta-toolchain-qt5-cortexa7hf-neon-toolchain-4.1.15-2.1.0.sh
```

直接执行脚本安装交叉编译工具，连续敲下两次回车键确认，再输入用户密码即可。本次安装的目录为脚本所指定的默认安装的目录，后面的内核编译环境的交叉编译都是按这个安装目录去操作，所以建议用户也是默认安装到/opt/fsl-imx-x11/4.1.15-2.1.0 这个默认目录。

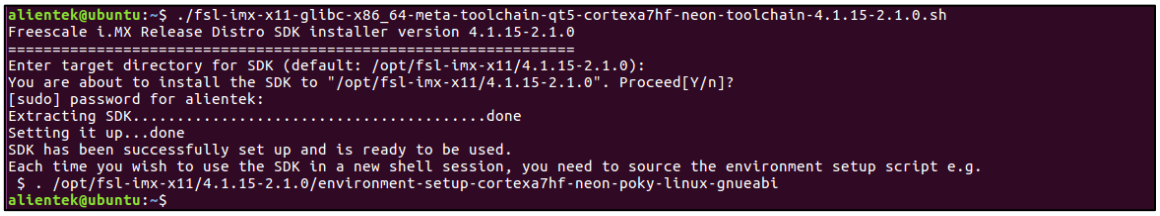


图 4.2.2 安装 sdk 工具到默认的目录

使用方法也十分简单，根据上面打印出来的提示，直接使能环境变量就可以了。但是在不同终端或者切换用户时需要重新使能环境变量方可使用。

```
source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

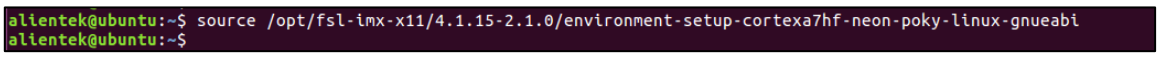


图 4.2.3 使能环境变量

使能环境变量后可以使用 env 指令查看生效的环境变量，下图为部分截图，可以看出使能了这个环境变量后 gcc 已经配置好编译时所用的参数，如硬浮点参数-mfpu=neon -mfloat-abi=hard。使用硬浮点交叉编译，可以使用 CPU 自带 FPU。下图为环境变量部分截图。

```
env
```

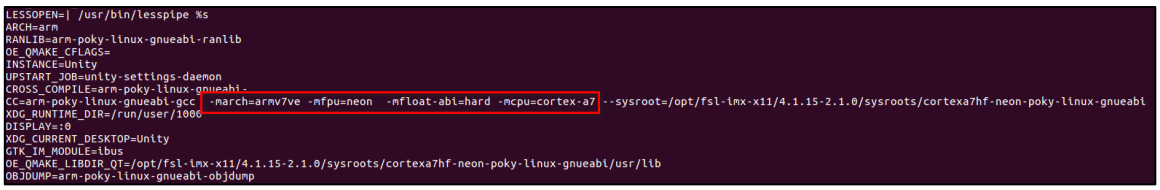


图 4.2.4 查看使能后的环境变量

使用 arm-poky-linux-gnueabi-gcc -v 指令可以查看 gcc 版本，表明环境变量已经生效。

```
arm-poky-linux-gnueabi-gcc --version
```

```
alientek@ubuntu:~$ arm-poky-linux-gnueabi-gcc --version
arm-poky-linux-gnueabi-gcc (GCC) 5.3.0
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

alientek@ubuntu:~$
```

图 4.2.5 查看 gcc 版本信息

要使用此编译器编译内核和 U-boot 还需要安装以下软件。

```
sudo apt-get update        // 先更新软列表
sudo apt-get install lzop  // 安装 lzop 工具，用于生成压缩或解压镜像
sudo apt-get install libncurses* // 安装 ncurese 相关库，U-boot 或者内核菜单显示时需要
```

4.3 编译出厂源码 U-boot

将开发板光盘 A-基础资料->1、例程源码->3、正点原子 Uboot 和 Linux 出厂源码->uboot-imx-2016.03-2.1.0-gxxxxxx-vxx.tar.bz（x 代表未版本，可能为.xz 格式或者.bz2 格式） U-boot 源码解压到 Ubuntu 拷贝到虚拟机 Ubuntu 家目录（当前用户目录）下，使用 tar 指令解压。
先在家目录创建一个 IMX6/uboot-imx-2016.03-2.1.0 目录，并将 U-boot 源码解压到此目录。

```
mkdir -p IMX6/uboot-imx-2016.03-2.1.0
tar xf uboot-imx-2016.03-2.1.0-gd3f0479-v1.4.tar.xz -C IMX6/uboot-imx-2016.03-2.1.0/
cd IMX6/uboot-imx-2016.03-2.1.0
ls
```

```
alientek@ubuntu:~$ mkdir -p IMX6/uboot-imx-2016.03-2.1.0
alientek@ubuntu:~$ tar xf uboot-imx-2016.03-2.1.0-gd3f0479-v1.4.tar.xz -C IMX6/uboot-imx-2016.03-2.1.0/
alientek@ubuntu:~$ cd IMX6/uboot-imx-2016.03-2.1.0
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$ ls
api   board     cmd      config.mk  disk  drivers  examples  include  Kconfig  Licenses     MAKEALL   net   README   snapshot.commit  tools
arch  build.sh  common   configs    doc   dts      fs        Kbuild   lib      MAINTAINERS  Makefile  post  scripts  test
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$
```

图 4.3.1 拷贝 U-boot 源码到 Ubtuntu 虚拟机

在编译 U-boot 之前，前提需要先安装 4.2 小节的 Poky 交叉编译工具链。

我们在 Linux 源码里写了一个脚本，脚本里已经配置好 deconfig 文件，和编译的目标文件。我们直接执行 build.sh，即可编译 U-boot 源码。编译的目标文件会在当前 U-boot 源码顶层目录下的 tmp 文件夹里。

```
build.sh        // 开始编译出厂 U-boot 源码，生成 u-boot.imx，第一级启动引导文件。
```

```
alientek@ubuntu:~/IMX6/uboot-imx-2016.03-2.1.0$ ./build.sh
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
```

图 4.3.2 执行 build.sh 脚本，编译 U-boot 源码

编译完成后查看当前目录下的 tmp 文件夹编译的文件目标，*imx 是已经添加头部信息的 U-boot 镜像，可直接使用 dd 指令烧写到 TF 卡和开发板上的 eMMC 储存设备，详细请参考【正点原子】I.MX6U 开发板文件拷贝及固件更新参考手册 V1.x.pdf；*bin 是未添加状信息的 U-boot 镜像，需要使用【正点原子】I.MX6U 嵌入式 Linux 驱动开发指南里所说的 imxdownload 工具烧写。

图 4.3 3 编译完成并查看编译的目标文件

4.4 编译出厂源码内核及模块

将开发板光盘 A-基础资料->1、例程源码->3、正点原子 Uboot 和 Linux 出厂源码->linux-imx-4.1.15-2.1.0-gxxxxxx-vxx.tar.xz（x 代表未知版本，可能为.xz 格式或者.bz2 格式）Linux 源码拷贝到 Ubuntu 拷贝到虚拟机 Ubuntu 家目录（当前用户目录）下，使用 tar 指令解压。

首先在家目录下创建 IMX6/linux-imx-4.1.15-2.1.0 目录，我们将 Linux 源码解压到此目录下。

```
mkdir -p IMX6/linux-imx-4.1.15-2.1.0
tar xf linux-imx-4.1.15-2.1.0-gb78e551-v1.4.tar.xz -C IMX6/linux-imx-4.1.15-2.1.0/
cd IMX6/linux-imx-4.1.15-2.1.0
ls
```

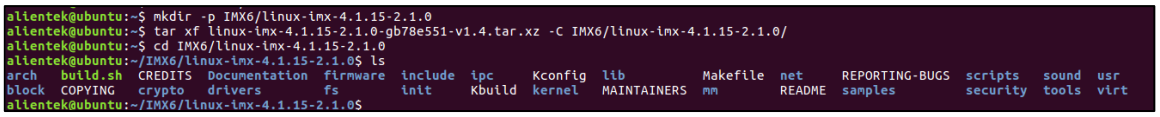


图 4.4 1 解压 Linux 源码后查看内核源码目录下的文件

在编译内核之前，前提需要先安装 4.2 小节的 Poky 交叉编译工具链。

我们在 Linux 源码里写了一个脚本，脚本里已经配置好 deconfig 文件，和编译的目标文件。我们直接执行 build.sh，即可编译 Linux 源码。编译的目标文件会在当前 Linux 源码顶层目录下的 tmp 文件夹里。

```
build.sh        // 开始编译出厂 Linux 源码，包括 zImage，dtb 和 modules。
```

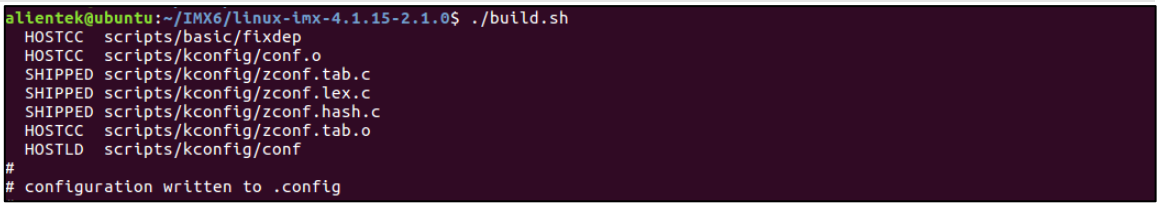


图 4.4 2 执行 build.sh 编译 Linux 源码

编译完成，及查看 tmp 目录下的编译目标文件，如下图，包含很多 dtb 文件（设备树），及 Linux 内核 zImage，还有 modules.tar.bz2（内核模块）。

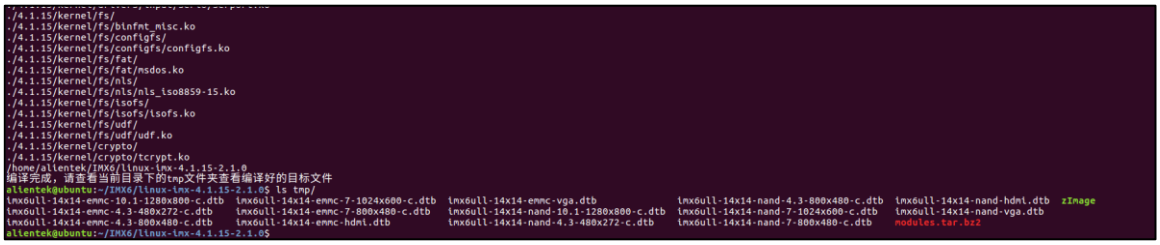


图 4.4 3 编译完成及查看编译后的目标文件

更新内核的方法请参考【正点原子】I.MX6U 开发板文件拷贝及固件更新参考手册 V1.x.pdf 文档。

4.6 编译出厂 Qt GUI 综合 Demo

在编译出厂 Qt GUI 之前，前提需要先安装 4.2 小节的 Poky 交叉编译工具链。

拷贝开发板光盘 A-基础资料->1、例程源码->9、Qt 综合例程源码下的 QDestkop 文件夹到 Ubuntu 虚拟机。如下图，编者已经拷贝到虚拟机家目录（用户目录）下，查看 QDesktop 目录下的文件如下。

```
alientek@ubuntu:~/QDesktop$ ls
aflex       alarms      desktop   helpbutton  main.cpp  media  photoview     qml.qrc              radio   settings  system     tcpserver  weather
aircondition  calculator  fileview  iotest      main.qml  music  QDesktop.pro  qtquickcontrols2.conf  sensor  src       tcpclient  udpchat    wireless
alientek@ubuntu:~/QDesktop$
```

图 4.6.1 拷贝 QDesktop 工程到 Ubuntu 虚拟机

使能第 4.2 小节的交叉编译环境变量。

```
source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

```
alientek@ubuntu:~/QDesktop$ source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
alientek@ubuntu:~/QDesktop$
```

图 4.6.2 使能交叉编译环境变量

执行 qmake，生成 Makefile，用于编译。

```
qmake
```

```
alientek@ubuntu:~/QDesktop$ qmake
Info: creating stash file /home/alientek/QDesktop/.qmake.stash
alientek@ubuntu:~/QDesktop$
```

图 4.6.3 生成 Makefile 用于编译

执行 make，开始编译 Qt 综合 Demo。

```
make -j 16    // -j 16 是允许最大同时 16 条编译任务同时执行，16 一般取分配虚拟机的核心数的 2 倍
```

编译过和中有报警告，可忽略。编译完成如下图，编译得到的 QDesktop 文件，可以直接拷贝到出厂系统下直接执行./QDestkop 即可。

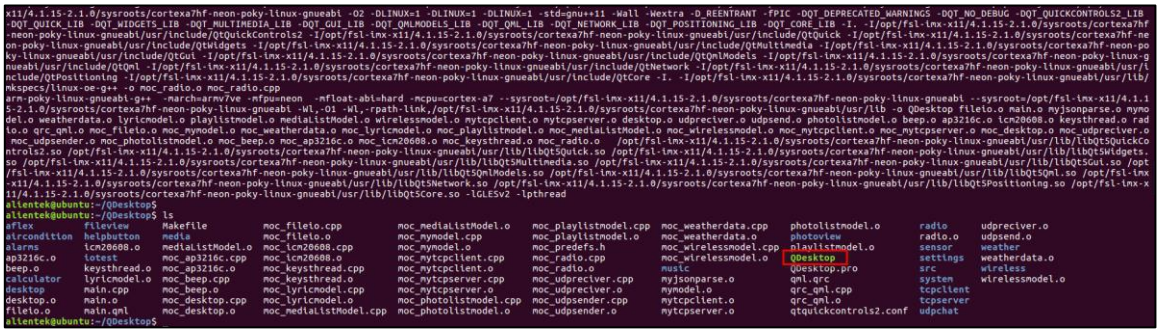


图 4.6.4 编译成功，生成 QDesktop 可执行文件

在开发板运行后的桌面如下，Qt Demo 的用法都在每个 App 里右上角的“关于”页面可查看。

图 4.6 5 出厂 GUI 第一屏演示

4.7 编译一个简单的 c 文件

在编译出厂 Qt GUI 之前，前提需要先安装 4.2 小节的 Poky 交叉编译工具链。

本文新建一个 test 文件夹，在 test 文件夹下编辑一个 main.c 文件，使用 vi/vim 指令，复制下面的内容。程序功能是打印一句“hello world!”，拷贝以下内容到 vi/vim 编辑器窗口里。按 i 进入插入模式，右键进行粘贴内容，粘贴成功后按 ESC 键退出编辑模式，然后输入:wq 保存并退出。

```
mkdir test
cd test
vi main.c
```

在 main.c 复制粘贴下面的内容。

```
#include <stdio.h>
int main(void)
{
        printf("hello world!\n");
        return 0;
}
```

如果您在上文已经使能过环境变量就不用再执行下面这一步了。

```
source /opt/fsl-imx-x11/4.1.15-2.1.0/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

编译 main.c 文件，指令解释

（1） $CC ：$是取值符号，取终端的环境变量 CC 的值.

（2） main.c ：c 文件

（3） -o ：参数-o，后面加编译的目标文件

```
$CC main.c -o main
```

或者写成下面这样来编译 main.c 文件也是可以的，实际是把$CC 的值换成了如下，下面为

一条指令，由于 PDF 格式问题，建议分段复制。

```
arm-poky-linux-gnueabi-gcc -march=armv7ve -mfpu=neon -mfloat-abi=hard -mcpu=cortex-a7 --sysroot=$SDKTARGETSYSROOT main.c -o main
```

```
alientek@ubuntu:~$ mkdir test
alientek@ubuntu:~$ cd test/
alientek@ubuntu:~/test$ vi main.c
alientek@ubuntu:~/test$ $CC main.c -o main
alientek@ubuntu:~/test$ ls
main  main.c
alientek@ubuntu:~/test$
```

图 4.7 1 编译 main.c

编译好的 main 文件，直接拷贝到出厂系统里，在串口终端执行./main，结果是串口打印一句“hello world!”。

第五章 ATK I.MX6U 文件系统功能简介

本章主要简介正点原子 I.MX6U 出厂文件系统的功能。

5.1 文件系统目录简介

ATK-IMX6U 出厂 Qt 文件系统目录简介:

目录	目录功能或放置的内容
/bin	系统放置了许多执行文件，如 mv、cp、date 等指令，这些指令一般是单用户维护模式使用，还可以被 root 用户使用
/boot	boot 目录一般放开机启动文件，但是 I.MX6U 已经有单独一个分区 boot 存储启动文件（zImage 和设备树 dtb 文件）
/dev	Linux 系统使用任何设备与接口都是以文件的形式存放在这个目录下，在 Linux 下一切皆文件
/etc	系统配置文件几乎都放在这个目录，例如启动脚本及所有的服务文件等
/home	默认用户主文件夹，默认存在 root 用户。新建的用户都会在此生成用户主文件夹
/lib	系统函数库，及内核模块 modules（驱动程序）
/mnt	一般用于临时挂载目录
/opt	第三方软件放置的目录，所以出厂 Qt 应用程序及所用到歌曲歌词等都放在这个目录下
/proc	此目录是一个虚拟的文件系统，它的数据在内存中，如内核、进程、外部设备和网络状态等，所以不占磁盘空间
/run	放置系统运行时所需要文件，以前防止在/var/run 中，后来拆分成独立的/run 目录。重启后重新生成对应的目录数据
/sbin	放置 sbin 目录下的指令是系统指令，只有 root 用户可以执行
/sys	与/proc 目录一样，是虚拟的文件系统，记录核心系统的硬件信息
/tmp	存放临时文件目录
/usr	Linux 核心目录，目录下包括非常多的文件，如共享文件，一些可执行文件等等
/var	存放系统执行过程经常改变的文件

5.2 查看 Qt 环境变量

在出厂文件系统里，常常把环境变量配置到/etc/profile 这个文件里。在/etc/profile 文件里配置的 Qt 环境变量（启动 Qt，需要配置 Qt 环境变量。），其中某些环境变量根据不同的显示设备需要配置或者取消配置相关环境变量，如下图。

```
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh ; do
    if [ -f $i -a -r $i ]; then
      . $i
    fi
  done
  unset i
fi

if [ -x /usr/bin/resize ];then
  # Make sure we are on a serial console (i.e. the device used starts with /dev/tty),
  # otherwise we confuse e.g. the eclipse launcher which tries do use ssh
  test "`tty | cut -c1-8`" = "/dev/tty" && resize >/dev/null
fi

export PATH PS1 OPIEDIR QPEDIR QTDIR EDITOR TERM
export LANG=en_US.UTF-8
export QT_QPA_FONTDIR=/usr/share/fonts/ttf
export QT_QPA_PLATFORM=linuxfb:tty=/dev/fb0
export QT_QPA_FB_TSLIB=1
umask 022

~
```

图 5.2 1 查看文件系统里 Qt 的环境变量配置

5.3 如何禁用 Qt 桌面启动

在出厂文件系统/etc/rc.local 文件里，如下图。不需要启动 Qt 界面，可以在/opt/QDesktop > /dev/null 2>&1 &行首前面加“#”注释掉这个指令即可。想要开启时再去除除“#”即可。

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.


echo 30000 >  /proc/sys/vm/min_free_kbytes
source /etc/profile
#/opt/QDesktop >/dev/null 2>&1 &
exit 0
```

图 5.3 1 查看出厂 Qt GUI 界面启动指令

5.4 如何创建自启动程序

与出厂 Qt 一样，自启动脚本/程序指令可以放到/etc/rc.local 这个文件里，因为/etc/rc.local 是系统启动最后一个执行的。当然也可以放在/etc/init.d/目录下的 rcS 文件里。

5.5 如何设置静态 ip

由于 LAN 8720 PHY 网络芯片没有 MAC 地址，系统启动时会生成随机 MAC 地址给网络使用。每次都生成一个 MAC 地址，导致每次开机网络获取的 ip 不是固定的。（v2.3 版本文件系统已固定一个随机 MAC 地址，不会因每次开机改变，请更新到最新文件系统。）所以我们要设置给它一个静态的 ip，不因 MAC 地址的改变而改变，设置固定 ip 方法调试。开发板启动时会自动获取 ip，最简单的方法是直接设置一个 ip 给它。这样我们就可以直接使用这个 ip 了。

在出厂文件系统里/etc/rc.local 文件里如图位置添加以下文件。

```
vi /etc/rc.local
```

添加以下内容，请根据个人使用的路由器，设置相应网段的 ip 地址。下图为设置 eth0 的静态 ip 地址。请注意这个 ip 与其他设备的 ip 冲突！

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
ifconfig eth0 192.168.1.115 netmask 255.255.255.0
route add default gw 192.168.1.1
echo "nameserver 114.114.114.114" > /etc/resolv.conf
```

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.


echo 30000 > /proc/sys/vm/min_free_kbytes
PATH=/sbin:/bin:/usr/sbin:/usr/bin
ifconfig eth0 192.168.1.115 netmask 255.255.255.0
route add default gw 192.168.1.1
echo "nameserver 114.114.114.114" > /etc/resolv.conf
source /etc/profile
/opt/QDesktop >/dev/null 2>&1 &
exit 0
```

图 5.5 1 快速设置一个静态 ip

5.6 系统常用软件版本

软件名称	软件版本	功能简介
Qt	Qt 5.12.9（2020.11.18 更新后）	支持 Qt Widgets 、Qt Quick、 Qt Charts、Qt Sqlite
Python	默认 python3.5（python2 需要额外安装，安装包位于开发板光盘 A-基础资料\08、系统镜像\01、出厂系统镜像\04、rpm 安装包）(文件系统版本 v2.4 以后)	运行 Python 程序
OpenCV	OpenCV 3.1	图像处理，调用摄像头可结合 Qt 使用等
GStreamer	Gstreamer1.8.1	媒体处理
Ffmpeg	ffmpeg version 3.0	音视频处理
Sqlite3	3.11.0	轻量级数据库
Nginx	1.8.1（2021.06.18 更新后）	轻量级 Web 服务器
…	…	…

至此，快速体验文档已经结束，感谢大家对正点原子的支持！

附录 A