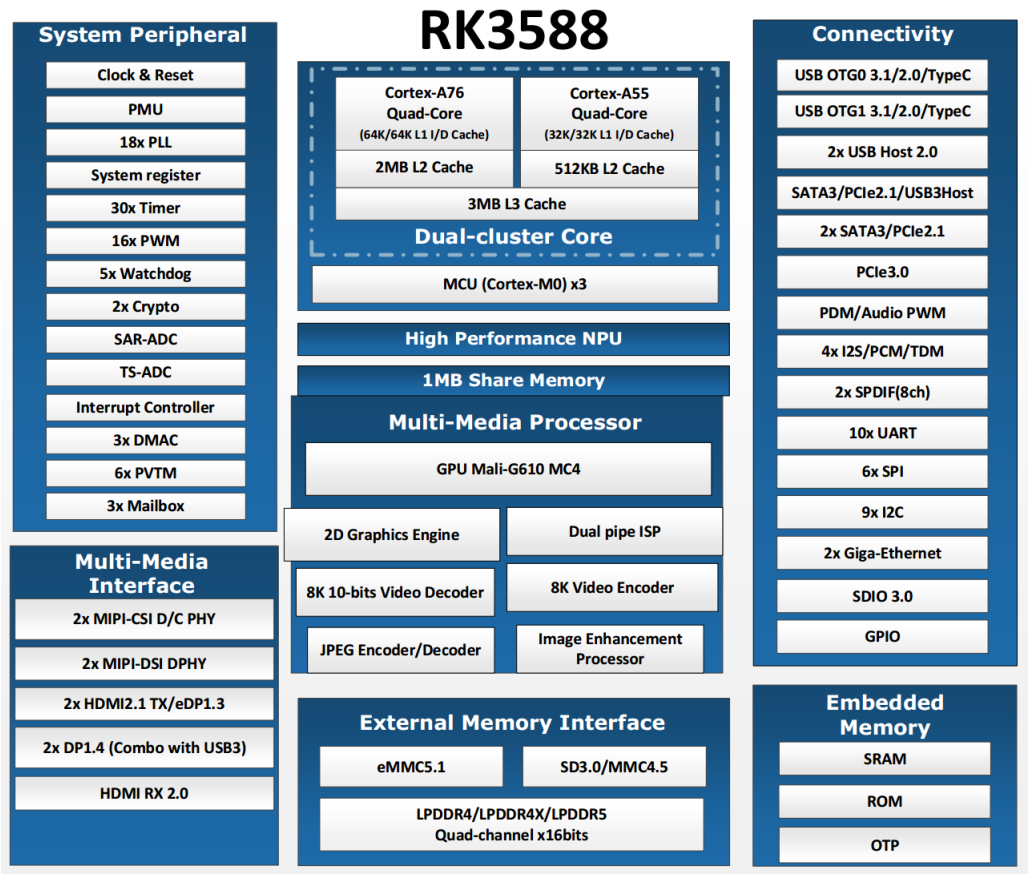
Rockchip RK3588 软件开发指导

RK3588 平台移植参考



苏仁义

me@email.cn

版本: 1.0

修订记录

版本变更	日期	作者	修改内容
1.1	2024/7/25	苏仁义	添加欧拉系统章节
1.0	2024/3/11	苏仁义	开始编写本文档

目录

第 I 部分

开发篇

1

代码编译

1.1 开发环境搭建

主机系统推荐使用 Ubuntu >= 20.04 的版本。

1.1.1 编译器和下载工具

从公司共享目录获取 RK3588 的编译工具：`\\192.168.1.202\常用工具\2. 开发工具\RK3588\gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz`。

下载后用放在 linux 系统下，用命令解压：

```
tar xpvf gcc-arm-9.2-2019.12-x86_64-aarch64-none-linux-gnu.tar.xz
```

下载 USB 的烧写程序和驱动程序文件：

`\\192.168.1.202\常用工具\2. 开发工具\RK3588\RkDevTool`。

其目录结构如下：

```
├── RKDevTool_Release_v2.96
│   ├── Language
│   ├── Log
│   └── bin
├── driver
└── rockdev
```

图 *1-1*: RkDevTool 目录

用于烧写的程序文件放在 rockdev 目录下面。双击

RKDevTool_Release_v2.96\RKDevTool.exe 程序，连接 USB 线来更新固件程序，运行界面如图 1-1 所示。

1.1.2 U-Boot 代码下载

U-Boot 源代码在这里：

http://192.168.1.202:8888/rk3588/uboot.git

用 git 命令下载：

```
git clone http://192.168.1.202:8888/rk3588/uboot.git
```

1.1.3 Kernel 代码下载

kernel 源代码在这里：

http://192.168.1.202:8888/rk3588/kernel.git

用 git 命令下载：

```
git clone http://192.168.1.202:8888/rk3588/kernel.git
```

准备好文件以后，目录结构如下图所示：

```
$ tree -d rk3588 -L 1
rk3588
├── gcc-arm-10.3-2021.07-x86_64-aarch64-none-linux-gnu
├── kernel
└── uboot
```

图 *1-2*: 代码目录结构

1.2 U-Boot

1.2.1 U-Boot 编译

用下面的命令编译 U-Boot：

```
./make.sh sytc
```

编译生成的文件是 uboot.img。

1.2.2 U-Boot 版本号

U-Boot 在启动的时候会打印我们内部的版本号，例如：

```
ON=0x00, OFF=0x00
vdd_gpu_s0 750000 uV
vdd_cpu_lit_s0 750000 uV
vdd_log_s0 750000 uV
vdd_vdenc_s0 init 750000 uV
vdd_ddr_s0 850000 uV
SYTC boot version: 1.0
get vp0 plane mask:0x5, primary id:2, cursor_plane:-1, from dts
get vp1 plane mask:0xa, primary id:3, cursor_plane:-1, from dts
get vp2 plane mask:0x140, primary id:8, cursor_plane:-1, from dts
get vp3 plane mask:0x280, primary id:9, cursor_plane:-1, from dts
Could not find baseparameter partition
Model: Rockchip RK3588 SYTC Board
```

图 *1-3*: U-Boot 内部版本号

给客户发布新版本的时候，需要修改版本号。这里有主、次版本号两个域。版本号的修改规则是：用于新的项目从 1.0 开始。主版本号与硬件的一致，硬件有升级，增加主版本号。软件的修改升级，需要增加次版本号。

U-Boot 的版本号在 `include/configs/sytc_rk3588.h` 里修改：

include/configs/sytc_rk3588.h:

```
#define SYTC_BOOT_MAJOR_VERSION 1
#define SYTC_BOOT_MINOR_VERSION 0
```

图 *1-4*: 修改 U-Boot 版本号

1.3 Kernel

1.3.1 kernel 编译

kernel 编译：

```
./make.sh
```

编译生成的文件是当前目录下的 boot.img。

1.3.2 kernel 版本号

为了内部版本管控，在内核代码的 `init/main.c` 中增加了内部版本号的打印：

init/main.c:start_kernel()

```
pr_notice("%s", linux_banner);
pr_notice("SYTC Linux version: %d.%d (%s)",
  CONFIG_SYTC_MAJOR_VERSION, CONFIG_SYTC_MINOR_VERSION, UTS_VERSION);
early_security_init();
setup_arch(&command_line);
```

图 *1-5*: Kernel 版本号

在启动过程中的打印中有内部版本号，例如：

```
[ 0.947325] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[ 0.947345] Linux version 5.10.110 (user@DESKTOP-8539AN4)
[ 0.947359] SYTC Linux version: 1.0 (#14 SMP Mon Mar 18 15:58:58 CST 2024)
[ 0.954751] Machine model: Rockchip RK3588 SYTC Board
[ 0.954811] earlycon: uart8250 at MMIO32 0x00000000feb50000 (options '')
[ 1.008442] printk: bootconsole [uart8250] enabled
```

图 *1-6*: Kernel 版本号示例

版本号的修改规则与 U-Boot 的一样。需要修改的地方在文件 `arch/arm64/configs/rk3588_sytc_defconfig`中：

arch/arm64/configs/rk3588_sytc_defconfig:

```
CONFIG_SYTC_MAJOR_VERSION=1
CONFIG_SYTC_MINOR_VERSION=0
```

图 *1-7*: 修改 Kernel 版本号

1.4 设备树文件

U-Boot 和 Kernel 使用的都是 boot.img 里面的 DTB 文件。设备树文件从这个文件：

arch/arm64/boot/dts/rockchip/rk3588-sytc.dts:

```
/dts-v1/;

/* #include "rk3588-sytc-hrgz.dtsi" */
#include "rk3588-sytc-evb.dtsi"

/ {
    model = "Rockchip RK3588 SYTC Board";
    compatible = "rockchip,rk3588-sytc", "rockchip,rk3588";

    chosen: chosen {
      bootargs = "earlycon=uart8250,mmio32,0xfeb50000 console=ttyFIQ0
        irqchip.gicv3_pseudo_nmi=0 root=PARTUUID=614e0000-0000 rw rootwait";
    };
};

&fiq_debugger {
    rockchip,baudrate = <115200>;
};
```

图 *1-8*: dts 入口文件

开始查看，它会通过 `include` 指令引入别的文件。新项目开始的时候建议从 `rk3588-sytc-evb.dtsi` 复制并重命名一个新文件，然后 `include` 它。如注释掉的 `rk3588-sytc-hrgz.dtsi` 一样。

1.5 rootfs.img 修改

rootfs.img 是一个 ext4 文件系统格式的文件，可以通过 mount 的方式挂载到系统下进行修改。由于这个文件给的剩余空间比较小，直接修改的方式可能会报失败。这里的示例是将文件先复制出来，按需修改好文件系统以后再重新做一个 ext4 的打包文件。

可以按下面的步骤进行修改：

(1) 将 rootfs.img 挂载到系统下，例如：

```
sudo mount -o loop rootfs.img /media
```

(2) 将 rootfs 的内容复制出来：

```
sudo mkdir rootfs
sudo cp -a /media/* rootfs/
sudo umount /media
```

(3) chroot 到 rootfs 的文件系统：

```
sudo mount -o bind /proc rootfs/proc
sudo mount -o bind /dev rootfs/dev
sudo mount -o bind /sys rootfs/sys
sudo chroot rootfs /bin/bash
```

(4) 安装新的软件包，例如：

```
apt-get update
apt-get install tcpdump
```

(5) 安装所需软件以后，清理临时文件并退出 chroot 环境：

```
apt-get clean
exit
```

(6) 卸载 bind 的文件系统：

```
sudo umount rootfs/proc
sudo umount rootfs/dev
sudo umount rootfs/sys
```

(7) 重新打包文件系统：

```
IMAGE_SIZE_MB=$(( $(sudo du -sh -m rootfs | cut -f1) + 300 ))
dd if=/dev/zero of=rootfs.img bs=1M count=0 seek=${IMAGE_SIZE_MB}
sudo mkfs.ext4 -d rootfs rootfs.img
```

文件系统里面的 root 用户密默认是 「root」。

1.6　打包升级文件

参考 RKDevTool_Release_v2.96\bin 目录下的 mkupdate.bat 脚本：

```
mklink /J Image ..\..\rockdev
Afptool.exe -pack ./ Image\update.img

RKImageMaker.exe -RK3588 Image\MiniLoaderAll.bin  Image\update.img update.img \
    -os_type:androidos

rem update.img is new format, Image\update.img is old format, so delete older one
del  Image\update.img

pause
```

打包进 update.img 在文件由 package-file 文件控制：

```
# NAME      Relative path
#
#HWDEF      HWDEF
package-file    package-file
bootloader  Image/MiniLoaderAll.bin
parameter   Image/parameter.txt
#
# if uboot.img is fit, uboot.img had include uboot and trust,
# so ignore trust.img
# file Image/uboot.img
# Image/uboot.img: Device Tree Blob version 17
#
# trust     Image/trust.img
#
uboot       Image/uboot.img
misc        Image/misc.img
#resource   Image/resource.img
#kernel     Image/kernel.img
boot        Image/boot.img
rootfs      Image/rootfs.img
recovery    Image/recovery.img
oem     Image/oem.img
userdata:grow   Image/userdata.img
# 要写入 backup 分区的文件就是自身（update.img）
# SELF 是关键字，表示升级文件（update.img）自身
# 在生成升级文件时，不加入 SELF 文件的内容，但在头部信息中有记录
# 在解包升级文件时，不解包 SELF 文件的内容。
#backup     RESERVED
#update-script  update-script
#recover-script recover-script
```

以 # 开头的行是注释。打包文件时要注意以下几点：

- 打包 update.img 固件时需要注意，升级固件不一定要全分区升级，可修改 package-file 文件，将不要升级的分区去掉，这样可以减少升级包（update.img ）的大小。
- package-file 中 recovery.img 如果打包进去的话，不会在 Recovery 模式中升级，为了预防升级 recovery.img 过程中掉电导致后面其他分区无法正常升级的问题，该分区升级放在 normal 系统下升级，即，执行 update 命令时会先检测 update.img 升级包中是否有打包 recovery.img，若有则升级 recovery 分区，再进入 Recovery 模式升级其他分区固件。

- misc 分区不建议打包进 update.img 中，即使有打包进去，也会在升级程序中加载判断到而忽略该分区，即使升级了 misc 分区，升级成功后 recovery 程序仍会清空 misc 分区中所有的命令及参数，从而导致达不到预想的结果。

- 如果将 update.img 升级包放置在 flash 中的 userdata 分区，则需要保证 package-file 中括不包括 userdata.img 被打包进去，原因是可能会导致文件系统的损坏，升级成功后可能使 oem 或 userdata 分区 mount 不成功。若从 SD 卡或 U 盘升级时，可以打包 userdata.img，从而对 userdata 分区进行升级。升级完成后会对 userdata 分区重新 resize 操作。

2

硬件接口

2.1 网络

系统里有两个网口，一个是从 SOC 直出的，一个从 PCIE 接 RTL81111 芯片出来的网口。他们的 IP 地址配置文件是 /etc/network/interfaces.d 下的 eth0 和 eth1。其对应关系如下表所示：

网络接口	驱动	默认地址
eth0	st_gmac *	192.168.1.80
eth1	r8168	192.168.2.80

表 *2.1*: 网口对应关系

网口的 MAC 地址参数会从 uboot 的环境变量中获取。但是 eth1 是从 pcie 出来的，没有 dts 设备节点，它的 MAC 地址需要处理。具体的方法请参看 `/etc/network/interface.d/eth1` 文件中的 pre-up 后的 script 文件。

*用命令 `ethtool -i eth0 | grep driver` 查看。

2.1.1 设备树节点

```
&gmac0 {
     /* Use rgmii-rxid mode to disable rx delay inside Soc */
     phy-mode = "rgmii-rxid";
     clock_in_out = "output";

     snps,reset-gpio = <&gpio4 RK_PB6 GPIO_ACTIVE_LOW>;
     snps,reset-active-low;
     /* Reset time is 20ms, 100ms for rtl8211f */
     snps,reset-delays-us = <0 20000 100000>;

     pinctrl-names = "default";
     pinctrl-0 = <&gmac0_miim
              &gmac0_tx_bus2
              &gmac0_rx_bus2
              &gmac0_rgmii_clk
              &gmac0_rgmii_bus
              >;

     tx_delay = <0x43>;
     /* rx_delay = <0x3f>; */

     phy-handle = <&rgmii_phy>;
     status = "okay";
 };
```

图 *2-1*: gmac0 节点

注意 `snps,reset-gpio` 这个 gpio，它是连接到 PHY 芯片的复位管脚上的，特别需要对照原理图检查一下。另外，如果在一些环境下不稳定的话可以尝试调整这里的 `tx_delay` 和 `rx_delay`。delay 的调整一般还需要改 phy-mode 这个参数：

phy-mode	含义
rgmii	tx,tx 都不调整
rgmii-id	tx, rx 都调整
rgmii-txid	只调整 tx delay
rgmii-rxid	只调整 rx delay

表 *2.2*: phy-mode 参数

2.1.2 PHY 寄存器读写调试

驱动提供了读写寄存器的接口，路径：/sys/bus/mdio_bus/devices/stmmac-0:01，其中 stmmac-0:01 表示 gmac0 的 PHY 地址是 1。

2.1.2.1 读寄存器

```
cat /sys/bus/mdio_bus/devices/stmmac-1\:01/phy_registers
```

2.1.2.2 写寄存器

例如, 往 reg0 写入 0x8100：

```
echo 00 0x8100 > /sys/bus/mdio_bus/devices/stmmac-1\:01/phy_registers
```

2.1.3 输出 25M 时钟

对 gmac0:

```
&gmac0 {
    /* Use rgmii-rxid mode to disable rx delay inside Soc */
   phy-mode = "rgmii-rxid";
    clock_in_out = "output";
    snps,reset-gpio = <&gpio4 RK_PB3 GPIO_ACTIVE_LOW>;
    snps,reset-active-low;
    /* Reset time is 20ms, 100ms for rtl8211f */
    snps,reset-delays-us = <0 20000 100000>;
+   pinctrl-names = "default";
+   pinctrl-0 = <&gmac0_miim
+            &gmac0_tx_bus2
+            &gmac0_rx_bus2
+            &gmac0_rgmii_clk
+            &gmac0_rgmii_bus
+            &eth0_pins>;
    tx_delay = <0x45>;
    /* rx_delay = <0x43>; */
    phy-handle = <&rgmii_phy>;
    status = "okay";
};
&mdio0 {
    rgmii_phy: phy@1 {
        compatible = "ethernet-phy-ieee802.3-c22";
        reg = <0x1>;
+       clocks = <&cru REFCLKO25M_ETH0_OUT>;
```

```
    };
};
```

对 gmac1:

```
&gmac1 {
    /* Use rgmii-rxid mode to disable rx delay inside Soc */
   phy-mode = "rgmii-rxid";
+   clock_in_out = "output";
    snps,reset-gpio = <&gpio3 RK_PB2 GPIO_ACTIVE_LOW>;
    snps,reset-active-low;
    /* Reset time is 20ms, 100ms for rtl8211f */
    snps,reset-delays-us = <0 20000 100000>;
+   pinctrl-names = "default";
+   pinctrl-0 = <&gmac1_miim
+            &gmac1_tx_bus2
+            &gmac1_rx_bus2
+            &gmac1_rgmii_clk
+            &gmac1_rgmii_bus
+            &eth1_pins>;    tx_delay = <0x45>;
    /* rx_delay = <0x43>; */
    phy-handle = <&rgmii_phy>;
    status = "okay";
};
&mdio1 {
    rgmii_phy: phy@1 {
        compatible = "ethernet-phy-ieee802.3-c22";
        reg = <0x1>;
+       clocks = <&cru REFCLKO25M_ETH0_OUT>;
    };
};
```

2.1.4 RGMII 延时扫描

通过 /sys/devices/platform/fe1b0000.ethernet 和 /sys/devices/platform/fe1c0000.ethernet 节点下的 phy_lb_scan 来扫描延时值。扫描前需要拔掉网线。

通过 `phy_lb_scan` 节点扫描到一个窗口，会得到一个中间坐标，需要使用千兆速度 1000 来扫描:

```
echo 1000 > phy_lb_scan
```

把扫描的结果配置到 dts 的 gmac0 或 gmac1 节点下 (tx-delay 和 rx-delay)。

2.2 Camera

RK3588 camera 资源硬件如下所示，拥有 2 路 DCPHY, 2 路 DPHY, 一路 DVP, 6 路 CSI HOST, 一个 vicap 控制器, 2 个 isp 控制器。其中 2 路 DPHY 可以分解成 4x2lane 的模式工作，如下图所示：

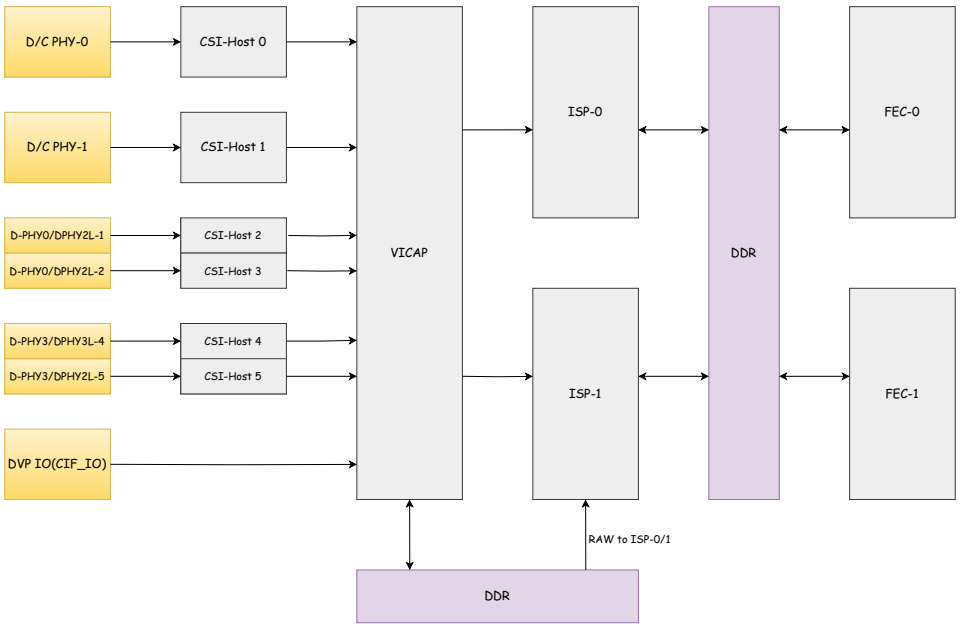


图 *2-2*: Camera 模块架构

2.2.1 Camera 概述

2.2.1.1 MIPI-CSI 资源

RK3588 MIPI-CSI 资源如下：

	Type	bandwidth	NUM	Mode
DPHY	DPHY-v1.2	2.5Gbps x 4 lanes	2	4lane or 2lane+2lane
DCPHY	DPHY-v2.0	DPHY-v2.0: 2.5Gbps x 4lanes MIPI D-PHY v1.2	2	4lane
	CSI-HOST	D-PHY v2.0 C-PHY v1.1	6	

RK3588 有 2 个 4lane 的 DPHY，2 个 4lane 的 DCPHY，其中 DPHY 可以拆分模式按照 4 个 2lane 进行工作，有 6 个 CSI-HOST 工作，总计可以接入 6 路 MIPI camera。2lane 最大带宽是 5G，分辨率可以达到 8M@30 帧，4lane 最大带宽达到 10G。

2.2.1.2 VICAP 资源

RK3588 VICAP 支持输入输出规格：

接口	数量	输入	输出
VICAP	1	BT601 YCbCr 422 8bit RAW8/10/12BT656 YCbCr 422 8bit 逐行/隔行 BT1120 YCbCr 422 16bit 逐行/隔行，单/双边沿采样 2/4 通道交错 BT656/BT1120 YCbCr 422 8/16bit 逐行/隔行 MIPI CSI 4 路 IDs 虚拟通道 MIPI CSI RAW8/10/12/14, YUV422	NV16/NV12/YUV400/YUYV 紧凑/非紧凑 RAW

2.2.1.3 ISP 资源

RK3588 的 ISP 属于 RK ISP v3.0 版本，拥有 2 个 ISP。

工作模式	吞吐率	最大分辨率	输入格式
单 ISP 单 CSI	16M@30fps	4672x3504	VICAP: raw8/raw10/raw12
单 ISP 2CSI	8M@30FPS	3840x2160/3264x2448	VICAP: raw8/raw10/raw12
单 ISP 4CSI	4M@30FPS	2560x1536	VICAP: raw8/raw10/raw12
双 ISP 2 合 1 单 CSI	32M@30fps 48M@15fps	8064x6048	VICAP: raw8/raw10/raw12

2.2.1.4 最多支持的 Camera

RK3588 最多支持 7 路正常的 CSI 摄像头。

MIPI-CSI	DVP 接口	支持 Camera 数量
6	1	1x48M@15fps 1x32M@30fps 2x16M@30FPS 4x8M@30FPS 7x4M@30FPS

RK3588 的 7 路 Camera 连接情况如下图所示：

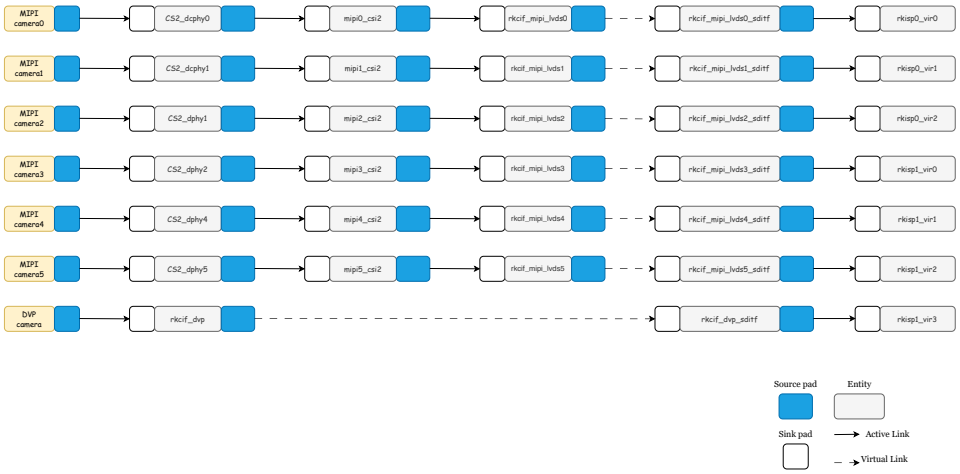


图 *2-3*: Camera 模块连接关系

根据上述的结构图可以大概看出 MIPI 和 DVP 分别是如何连接的，关键点如下：

(1) rk3588 支持两个 dcphy，节点名称分别为 csi2_dcphy0/csi2_dcphy1。每个 dcphy 硬件支持 RX/TX 同时使用，对于 camera 输入使用的是 RX。支持 DPHY/CPHY 协议复用；需要注意的是同一个 dcphy 的 TX/RX 只能同时使用 DPHY 或同时使用 CPHY。

(2) rk3588 支持 2 个 dphy 硬件，这里我们称之为 dphy0_hw/dphy1_hw，两个 dphy 硬件都可以工作在 full mode 和 split mode 两种模式下。

- full mode：节点名称使用 csi2_dphy0 和 csi2_dphy3，最多支持 4 lane。
- split mode：拆分成 2 个 phy 使用，分别为 csi2_dphy1（使用 0/1 lane）、csi2_dphy2(使用 2/3 lane)，dphy1_hw 则拆分成 csi2_dphy4 和 csi2_dphy5，每个 phy 最多支持 2 lane。
- 当 dphy0_hw 使用 full mode 时，链路需要按照 csi2_dphy1 这条链路来配置，但是节点名称 csi2_dphy1 需要修改为 csi2_dphy0，软件

上是通过 phy 的序号来区分 phy 使用的模式。dphy1_hw 同理。

(3) 使用上述 mipi phy 节点，需要把对应的物理节点 csi2_dcphy0_hw/csi2_dcphy1_hw/csi2_dphy0_hw/csi2_dphy1_hw 配置上。

(4) 每个 mipi phy 都需要一个 csi2 模块来解析 mipi 协议，节点名称分别 mipi0_csi2 ~mipi5_csi2。

(5) rk3588 所有 camera 数据都需要通过 vicap，再链接到 isp。rk3588 仅支持一个 vicap 硬件，这个 vicap 支持同时输入 6 路 mipi phy，及一路 dvp 数据，所以我们将 vicap 分化成 rkcif_mipi_lvds ~rkcif_mipi_lvds5、rkcif_dvp 等 7 个节点，各个节点的绑定关系需要严格按照框图的节点序号配置。

(6) 每个 vicap 节点与 isp 的链接关系，通过对应虚拟出的 XXX_sditf 来指明链接关系。

(7) rk3588 支持 2 个 isp 硬件，每个 isp 设备可虚拟出多个虚拟节点，软件上通过回读的方式，依次从 ddr 读取每一路的图像数据进 isp 处理。对于多摄方案，建议将数据流平均分配到两个 isp 上。

(8) 直通与回读模式：

- 直通：指数据经过 vicap 采集，直接发送给 isp 处理，不存储到 ddr。需要注意的是 hdr 直通时，只有短帧是真正的直通，长帧需要存在 ddr，isp 再从 ddr 读取。
- 回读：指数据经过 vicap 采集到 ddr，应用获取到数据后，将 buffer 地址推送给 isp，isp 再从 ddr 获取图像数据。
- dts 配置时，一个 isp 硬件，如果只配置一个虚拟节点，默认使用直通模式，如果配置了多个虚拟节点默认使用回读模式。

2.2.1.5 AHD 摄像头

AHD 摄像头可通过转换芯片，转成支持 4 路虚拟通道的 MIPI/BT1120 接口到 RK3588 最大可以支持到 28 路 1080p30 的摄像头。

MIPI-CSI	DVP 接口	支持 Camera 数量
6	1	28(7x4)*1920x1080@30FPS

2.2.2 RTP Stream

从 HDMI 输入的视频数据，在 Linux 系统里从 /dev/videoX 的设备中获取出来。可以用多媒体框架 gstreamer 来做测试。用 gstreamer 的 gst-launch-1.0 可以用来控制 video 的视频数据。

这里以板子上用 gstreamer 提供 rtp 服务，在电脑的 windows 端用 VLC 程序来实现视频的播放为例，展示一下 video 功能。

(1) 在板子上启动 rtp 服务：

```
gst-launch-1.0 v4l2src device=/dev/video0 ! videoconvert ! x264enc \
    ! rtph264pay pt=96 ! udpsink host=192.168.1.27 port=5000
```

用 RockChip 的 H264 硬编码，可以用下面命令：

```
gst-launch-1.0 v4l2src device=/dev/video0 ! mpph264enc ! rtph264pay pt=96 \
    ! udpsink host=192.168.1.27 port=5000
```

这里假定电脑的 IP 地址是：192.168.1.27。这根据实际情况修改。

(2) 在电脑上新建一个名为 hdmi.sdp 的 sdp 文件*，内容如下：

```
m=video 5000 RTP/AVP 96

a=rtpmap:96 H264

a=framerate:30

c=IN IP4 192.168.1.27
```

这里的 5000 是端口号，192.168.1.27 是 IP 地址，要与上面命令中的参数保持一致。

(3) 在电脑上用 VLC 程序打开 hdmi.sdp 文件：

*SDP 的完整定义，请查看 rfc4566

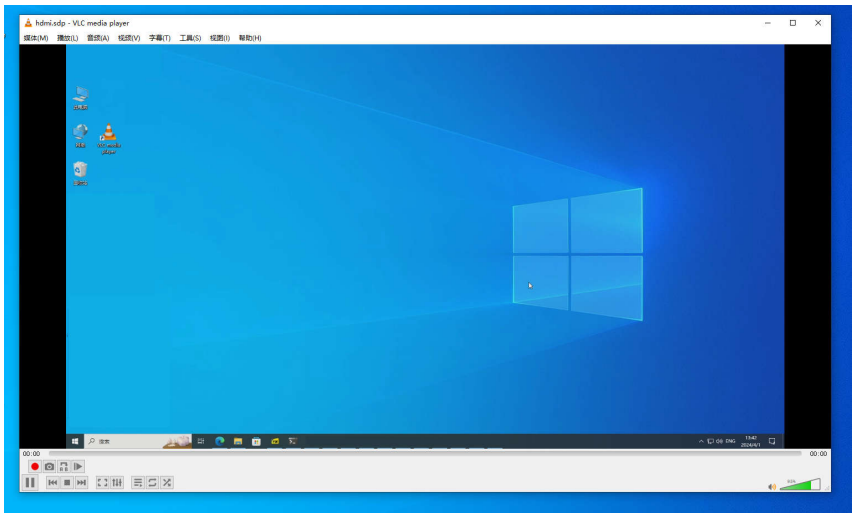


图 *2-4*: VLC 播放 video

如果一切正常可以在 vlc 里面看到 HDMI 输入的视频。

注意这里的操作顺序，最好是打开 vlc, 紧接着运行 gs-launch 命令，否则会出现很久才出视频。

2.2.3 V4L2 调试

(1) 推灰度图像的视频流：

```
gst-launch-1.0 v4l2src device=/dev/video1 ! capsfilter        \
    caps=video/x-raw,format=GRAY8 ! videoconvert ! mpph264enc \
    ! rtph264pay config-interval=5 pt=96 !                    \
    udpsink host=192.168.1.27 port=5000
```

(2) 将灰度图像显示到屏上：

```
gst-launch-1.0 v4l2src device=/dev/video1 io-mode=dmabuf      \
    ! capsfilter caps=video/x-raw,format=GRAY8 ! videoconvert \
    ! capsfilter caps=video/x-raw,format=RGB ! kmssink
```

(3) 用 dmabuf 模式并推流：

```
gst-launch-1.0 v4l2src device=/dev/video0 io-mode=dmabuf ! \
    mpph264enc ! rtph264pay config-interval=5 pt=96 !      \
    udpsink host=192.168.1.27 port=5000
```

(4) 将 camera 视频显示在屏上：

```
gst-launch-1.0 v4l2src device=/dev/video0 io-mode=dmabuf ! kmssink \
    force-modesetting=true
```

(5) 抓图：

```
v4l2-ctl -d 0 --stream-mmap --stream-count=1 --stream-to file.yuv
```

3

CPU、DDR 和 NPU 频率

通常，板子上的各个单元的频率是动态调频，这种情况下测试出来的性能会有波动。为了防止性能测试结果不一致，在性能评估时，建议固定板子上的相关单元的频率再做测试。

3.1 CPU 定频命令

3.1.1 查看 CPU 频率

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

3.1.2 固定 CPU 频率

查看可用的频率

```
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
```

可能的输出如下 (单位是 kHZ):

```
408000 600000 816000 1008000 1200000 1416000 1608000 1800000
```

设置 CPU 频率，如 1.8G

```
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
echo 1800000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
```

3.2 DDR 定频命令

3.2.1 查看 DDR 频率

```
cat /sys/class/devfreq/dmc/cur_freq
```

3.2.2 固定 DDR 频率

查看 DDR 可用的频率

```
cat /sys/class/devfreq/dmc/available_frequencies
```

可能的输出如下

```
528000000 1068000000 1560000000 2112000000
```

设置 DDR 频率，例如，2112000000 HZ

```
echo userspace > /sys/class/devfreq/dmc/governor
echo 2112000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

3.3 NPU 相关

3.3.1 查看 NPU 频率

用下面命令查看：

```
cat /sys/kernel/debug/clk/clk_summary | grep clk_npu_dsu0
```

可能的输出如下:

```
    clk_npu_dsu0    3     6     0    250000000     0      0  50000
```

3.3.2 设置 NPU 频率 (需要固件支持)

例如，设置为 1GHz

```
echo 1000000000 > /sys/kernel/debug/clk/clk_npu_dsu0/clk_rate
```

3.3.3 查询 NPU 驱动版本

```
[root@openEuler ~]# cat /sys/kernel/debug/rknpu/version
RKNPU driver: v0.8.2
```

3.3.4 查询 NPU 电源状态

```
cat /sys/kernel/debug/rknpu/power
```

3.3.5 打开 NPU 电源

```
echo on > /sys/kernel/debug/rknpu/power
```

3.3.6 关闭 NPU 电源

```
echo off > /sys/kernel/debug/rknpu/power
```

4

U-Boot 快捷键

RK 平台提供串口组合键触发一些事件用于调试、烧写（如果无法触发，请多尝试几次；启用 secure-boot 时无效）。**开机时长按：**

- `ctrl+c`：进入 U-Boot 命令行模式；
- `ctrl+d`：进入 loader 烧写模式；
- `ctrl+b`：进入 maskrom 烧写模式；
- `ctrl+f`：进入 fastboot 模式；
- `ctrl+m`：打印 bidram/system 信息；
- `ctrl+i`：使能内核 initcall_debug；
- `ctrl+p`：打印 cmdline 信息；
- `ctrl+s`：“Starting kernel...” 之后进入 U-Boot 命令行；

5

文件更新

5.1 MMC 分区

默认情况下，MMC 的分区如下表所示：

起始 (扇区)	大小 (字节)	分区名	程序
0x0000000	4M	loader	MiniLoaderAll.bin
0x0002000	1M	bootenv	uboot 环境变量
0x0004000	8M	uboot	uboot.img
0x0008000	80M	boot	boot.img
0x0030000	14G	rootfs	rootfs.img
0x1c30000	-	userdata	userdata.img

表 *5.1*: MMC 分区表

这里一个扇区的大小是 512 字节。

5.2 U-Boot 网络升级

开机时在调试串口上按住 「CTRL-C」，串口打印将停止在 U-Boot 的命令行上。

5.2.1 升级 uboot

```
tftpflash 20000000 uboot.img uboot
```

如果下载失败，请检查网线是否接在 CPU 直出的网口上面。

5.2.2 升级 boot

```
tftpflash 20000000 boot.img boot
```

如果下载失败，请检查网线是否接在 CPU 直出的网口上面。

5.2.3 升级 rootfs

```
tftpflash 20000000 rootfs.img rootfs
```

注意：rootfs 文件比较大，uboot 下可用的内存 3GB 左右，文件先下到内存会放不下而失败。

6

gstreamer 用法示例

6.1 推 MJPEG 的视频流

一般 USB 接口的摄像头支持 Motion JPEG 的视频流。可以用下面的命令来推 RTP 的视频流：

```
gst-launch-1.0 v4l2src device=/dev/video0 ! 'image/jpeg,width=1280,height=720,\
  framterate=30/1' ! rtpjpegpay ! udpsink host=192.168.1.27 port=5000
```

这里假定接收端的 IP 地址是 '192.168.1.27'。接收端用 VLC 软件，需要配置一个 sdp 文件, 内容如下：

```
m=video 5000 RTP/AVP 26
a=rtpmap:26 JPEG/90000
a=framerate:30
c=IN IP4 192.168.1.27
```

摄像头支持的视频格式，可以用命令进行查看：

```
v4l2-ctl -d 0 --list-formats-ext
```

这里的 '0' 对应 '/dev/video0'，也可以写成 '/dev/video0' 的形式。

6.2 推 TS 流

用 gstreamer 推 RTP 的视频流，用 VLC 一般需要一个 sdp 文件才能正常播放。推 ts 的流，VLC 一般可以直接播放。例如，用组播推 ts 流：

```
gst-launch-1.0 v4l2src device=/dev/video0 ! 'image/jpeg,width=1280,height=720,\
  framterate=30/1' ! jpegdec ! mpph264enc ! mpegtsmux ! udpsink host=224.2.2.5 \
  port=5000 ttl-mc=225 ttl=255
```

VLC 打开串流: “udp://@224.2.2.5:5000”。

用 udp 单播推 ts 流（这里以目标地址 ‘192.168.1.27’, 端口 ‘5000’ 为例）:

```
gst-launch-1.0 v4l2src device=/dev/video20 ! 'image/jpeg,width=1280,height=720,\
  framerate=30/1' ! jpegdec ! mpph264enc ! h264parse config-interval=10 ! \
  mpegtsmux ! udpsink host=192.168.1.27 port=5000
```

在目标机上，用 VLC 打开串流：“udp://@:5000”。

7

常见问题

7.1 U-Boot 打印 Overflow

```
sysmem_dump_all:
    ----------------------------------------------------------------------
    memory.rgn[0].addr     = 0x00200000 - 0x08400000 (size: 0x08200000)
    memory.rgn[1].addr     = 0x09400000 - 0xf0000000 (size: 0xe6c00000)

    memory.total           = 0xeee00000 (3822 MiB. 0 KiB)
    ----------------------------------------------------------------------
    allocated.rgn[0].name  = "UBOOT"
                    .addr  = 0xeb7fac30 - 0xf0000000 (size: 0x048053d0)
    allocated.rgn[1].name  = "STACK"     <Overflow!>
                    .addr  = 0xeb67ac30 - 0xeb7fac30 (size: 0x00180000)
    allocated.rgn[2].name  = "FDT"
                    .addr  = 0x0a100000 - 0x0a12b404 (size: 0x0002b404)

    kmem-resv.rgn[0].name  = "cma"
                    .addr  = 0x10000000 - 0x18000000 (size: 0x08000000)
    kmem-resv.rgn[1].name  = "ramoops@110000"
                    .addr  = 0x00110000 - 0x00200000 (size: 0x000f0000)

    framework malloc_r     =  32 MiB
    framework malloc_f     = 512 KiB

    allocated.total        = 0x0caa07d4 (202 MiB. 641 KiB)
    ----------------------------------------------------------------------
    LMB.allocated[0].addr  = 0x0a100000 - 0x0a12b404 (size: 0x0002b404)
    LMB.allocated[1].addr  = 0xeb67ac30 - 0xf0000000 (size: 0x049853d0)

    reserved.core.total    = 0x049b07d4 (73 MiB. 705 KiB)
    ----------------------------------------------------------------------
```

图 *7-1*：U-Boot Overflow

解决办法是将 U-Boot 的 lib/Kconfig 文件中的 `SYS_STACK_SIZE`，将其增大到 4MiB。这里的默认配置是 2MiB：

lib/Kconfig:

```
config SYS_STACK_SIZE
    hex
    default 0x200000   改这里
    help
        The system stack size.
```

图 7-2: 修改 U-Boot 栈大小

7.2　网卡名固定成 ethX

网卡接口名称默认会根据文件：/lib/udev/rules.d/80-net-setup-link.rules 中定义的规则来。如果你要更改规则，需要先将文件：80-net-setup-link.rules 从 /lib/udev/rules.d 目录复制到 /etc/udev/rules.d 目录。因为 /etc/udev/rules.d 目录下规则的优先级高于 /lib/udev/rules.d 目录，识别网卡并命名时，会优先从 /etc/udev/rules.d 目录下寻找规则文件。将 ID_NET_NAME 改成 ID_NET_SLOT 即可。

7.3　rc.local 脚本没执行

编辑/usr/lib/systemd/system/rc-local.service 文件，添加以下内容：

```
[Install]
WantedBy=multi-user.target
```

然后执行下面命令开启rc-local.service 服务：

```
systemctl enable rc-local.service
```

第 II 部分

生产部署

1

系统安装

电脑需要安装瑞芯微的 USB 驱动程序。接上 USB 线，运行 RKDevTool.exe 程序，然后上电进入 maskrom 模式，按下面步骤进行操作：

(1) 运行 RKDevTooL.exe 出现如下界面：

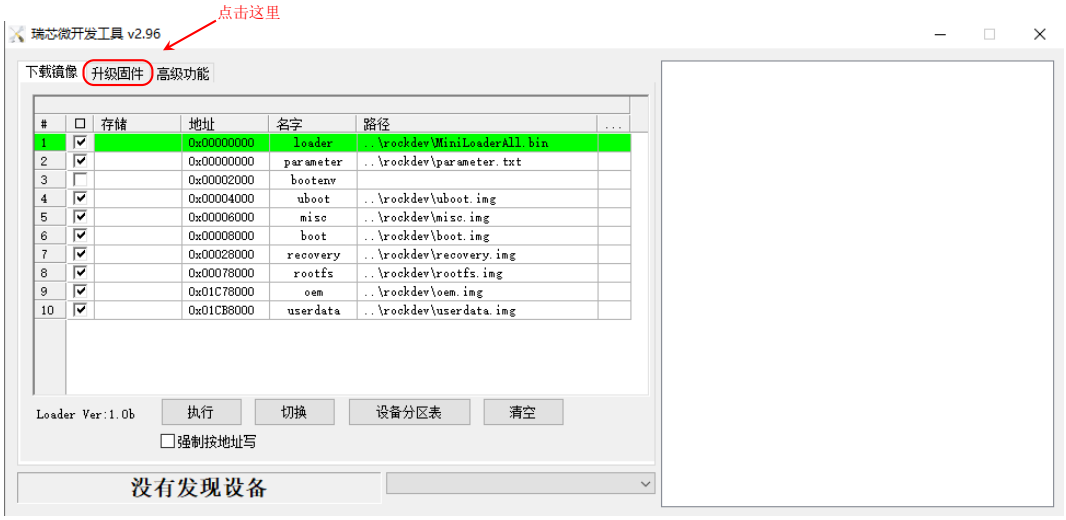
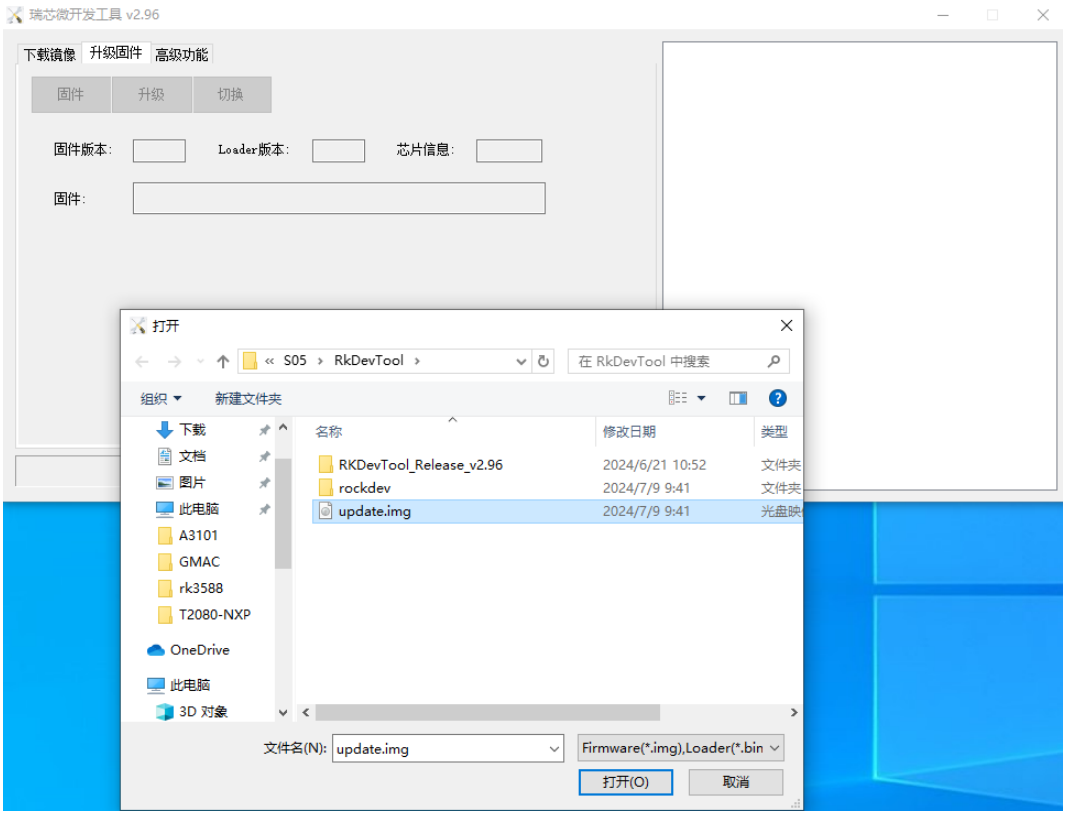


图 *1-1*：RKDevTooL 界面

(2) 点击上图中的“升级固件”:

点击 “固件” 按钮。

(3) 在弹出的对话框中选择正确的 “update.img” 文件, 等待 “打开” 执行完成。



(4) 点击 “升级” 按钮：等待升级完成。

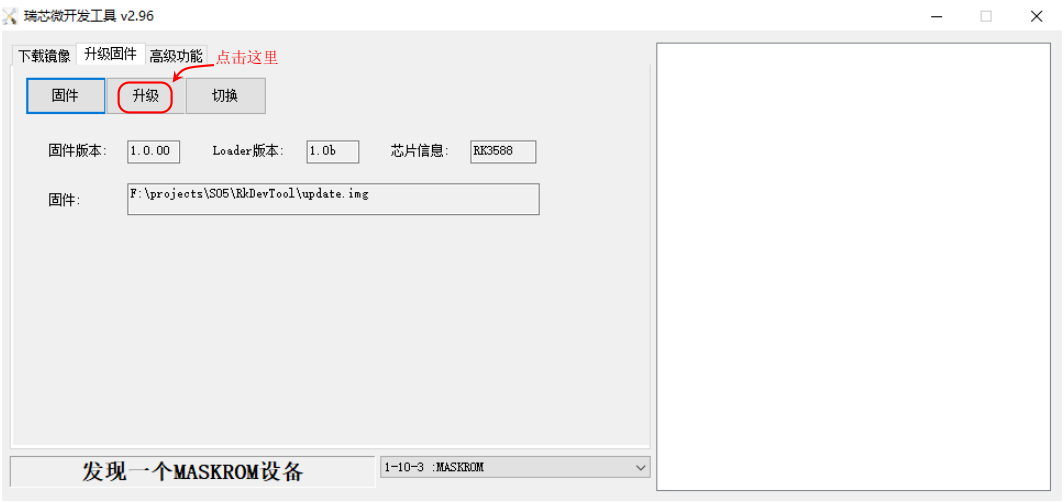


图 *1-2*: 升级 update.img

等待升级完成。

2

配置系统

在调试串口上按住 「CTRL-C」并上电开机，将调试串口停止在 U-Boot 的命令行上，用 「setsn」，将板卡的编号设置进去，例如：

```
=> setsn A2246002
Generate  ethaddr: 72:a2:24:60:02:01
Generate eth1addr: 72:a2:24:60:02:02
Generate eth2addr: 72:a2:24:60:02:03
Generate eth3addr: 72:a2:24:60:02:04
Writing to MMC(0)... done
```

图 *2-1*: 设置板卡编号

根据编号自动生成一组 mac 地址，并保存在环境变量里面。

将板卡名称设置为项目名并保存，例如：

```
=> setenv board_name SY-3588J01A
=> saveenv
Saving Environment to MMC...
Writing to MMC(0)... done
```

图 *2-2*: 设置板卡名称

第 III 部分

其他

1

WSL2 安装配置

1.1 WSL2 系统导入导出

在 powershell 或者 cmd 窗口进行 wsl2 系统的导入导出：

(1) 查看子系统是否运行：

```
wsl -l --running
```

(2) 终止子系统运行：

```
wsl -t < 子系统名 >
```

(3) 备份子系统：

```
wsl --export < 子系统名 > < 保存路径 >
```

例如：

```
wsl --export Ubuntu-22.04 e:\ubuntu-22.04-wsl2.zip
```

(4) 注销子系统：

```
wsl --unregister < 子系统名 >
```

(5) 恢复子系统:

```
wsl --import < 子系统名 > < 子系统安装路径 > < 子系统保存路径 >
```

例如：

```
wsl --import Ubuntu-22.04 e:\wls2\ubuntu2204 e:\ubntu-22.04-wsl2.zip
```

(6) 修改默认登录用户：

```
wsl --distribution Ubuntu-22.04 --user <user>
```

或者：

```
< 子系统名 >.exe config --default-user < 默认用户名 >
```

例如：

```
ubuntu2204.exe config --default-user < 默认用户名 >
```

1.2　vhd 文件挂载

1.2.1　Windows 下用 WSL 工具

使用 Microsoft Store 中的 WSL 可以直接挂载：

```
wsl --mount --vhd <pathToVHD>
```

例如：

```
wsl --mount --vhd e:\wsl2\rk3588\rk3588.vhdx --partition 1 --name rk3588
```

1.2.2　linux 下用 qemu-nbd

安装 qemu-nbd 工具：

```
apt-get install qemu-utils
```

加载 nbd 驱动：

```
modprobe nbd nbds_max=64
```

用 qemu-nbd 加载：

```
qenum-nbd -c /dev/nbd0 /path/to/vhdx
```

接下来就可以用 parted 工具查看或修改分区，用 mount 挂载文件系统等，总之就把 /dev/nbd0 当做一个磁盘进行使用。

1.3　USB 设备共享

在 windows 上安装 usbipd-win。使用方法可以参考这里：https://learn.microsoft.com/zh-cn/windows/wsl/connect-usb。

共享 usb 设备可以按以下步骤进行：

(1) 通过以管理员模式打开 PowerShell 并输入以下命令，列出所有连接到 Windows 的 USB 设备。列出设备后，选择并复制要附加到 WSL 的设备总线 ID：

```
usbipd list
```

(2) 在附加 USB 设备之前，必须使用命令 usbipd bind 来共享设备，从而允许它附加到 WSL。这需要管理员权限。选择要在 WSL 中使用的设备总线 ID，然后运行以下命令。运行命令后，请再次使用命令 usbipd list 验证设备是否已共享：

```
usbipd bind --busid 2-1
```

(3) 若要附加 USB 设备，请运行以下命令。（不再需要使用提升的管理员提示。）确保 WSL 命令提示符处于打开状态，以使 WSL 2 轻型 VM 保持活动状态。请注意，只要 USB 设备连接到 WSL，Windows 将无法使用它。附加到 WSL 后，任何作为 WSL 2 运行的分发版本都可以使用 USB 设备。使用 usbipd list 验证设备是否已附加。在 WSL 提示符下，运行 lsusb 以验证 USB 设备是否已列出，并且可以使用 Linux 工具与之交互：

```
usbipd attach --wsl --busid 2-1
```

(4) 打开 Ubuntu（或首选的 WSL 命令行），使用以下命令列出附加的 USB 设备：

```
lsusb
```

(5) 在 WSL 中完成设备使用后，可物理断开 USB 设备，或者从 PowerShell 运行此命令：

```
usbipd detach --busid <busid>
```

1.4　Windows 端口转发

按下面的步骤进行设置：

(1) 管理员权限启动 windows powershell，运行如下代码。这里以设置 80 端口转发 WSL 的地址的 80 端口为例：

```
netsh interface portproxy add v4tov4 listenport=80 \
    listenaddress=0.0.0.0 connectport=80 \
    connectaddress=172.19.11.104 protocol=tcp
```

(2) 查看端口转发情况：

```
netsh interface portproxy show all
```

(3) 删除端口转发：

```
netsh interface portproxy delete v4tov4 listenport=80  \
    listenaddress=0.0.0.0
```

2

制作 ubuntu 文件系统

在主机的 Ubuntu 系统里面，可以制作 aarch64 的文件系统，这里介绍一下步骤。

2.1 准备安装环境

2.1.1 安装依赖软件

```
sudo apt install qemu-user-static debootstrap
```

2.1.2 下载基础系统

这里以安装 Ubuntu 22.04 为例：

```
mkdir ubuntu
sudo debootstrap --arch=arm64 --foreign jammy ubuntu
sudo cp /usr/bin/qemu-aarch64-static ubuntu/usr/bin
sudo chroot ubuntu /usr/bin/qemu-aarch64-static /bin/sh \
    -i /debootstrap/debootstrap --second-stage
```

Ubuntu 22.04 的代号是 jammy, Ubuntu 20.04 的代号是 focal。如果要做 Ubuntu 20.04 版本的系统，把命令中的 jammy 换成 focal。

2.1.3 切换到安装环境

```
  sudo mount -t sysfs sys /sys ubuntu/sys
  sudo mount -t proc proc /proc ubuntu/proc
  sudo mount -o bind /dev ubuntu/dev
  sudo mount -o bind /dev/pts ubuntu/dev/pts
  sudo chroot ubuntu /usr/bin/qemu-aarch64-static /bin/bash -i
```

2.2 构建 rootfs

2.2.1 更新 apt 源

修改 /etc/apt/sources.list 如下：

```
deb http://ports.ubuntu.com/ jammy main universe
deb-src http://ports.ubuntu.com/ jammy main universe
deb http://ports.ubuntu.com/ jammy-security main universe
deb-src http://ports.ubuntu.com/ jammy-security main universe
deb http://ports.ubuntu.com/ jammy-updates main universe
deb-src http://ports.ubuntu.com/ jammy-updates main universe
```

2.2.2 更新系统

```
apt-get update
apt-get dist-upgrade
```

2.2.3 安装必要软件

```
apt-get install -y vim-nox build-essential gdb u-boot-tools rsync \
  socat jq tcpdump ifupdown minicom i2c-tools lrzsz tftp-hpa \
  net-tools dosfstools pciutils memtool ethtool unzip python3-pip \
  parted usbutils command-not-found gdisk openssh-server
apt-get install locales tzdata
```

2.2.4 安装 rockchip 硬解码器

添加软件源：

```
apt-get install software-properties-common
add-apt-repository ppa:george-coolpi/multimedia
apt update
```

安装 gstreamer ：

```
apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev \
libgstreamer-plugins-bad1.0-dev gstreamer1.0-plugins-base           \
gstreamer1.0-plugins-good gstreamer1.0-plugins-bad  alsa-utils      \
gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-tools     \
gstreamer1.0-alsa gstreamer1.0-pulseaudio ffmpeg v4l-utils
```

安装插件 gstreamer1.0-rockchip

```
apt-get install gstreamer1.0-rockchip
```

2.2.5 安装图形界面 (可选)

2.2.5.1 安装 ubuntu 桌面环境

执行如下命令：

```
apt-get install ubuntu-desktop # 其他可选 xubuntu-desktop lubuntu-desktop 等
```

提示选择桌面管理器时，可以选 ‘gdm‘。

2.2.5.2 安装中英文语言包与输入法 (可选)

```
# 英文环境
apt-get install language-pack-en-base
apt-get install language-pack-gnome-en-base
```

```
# 中文环境
apt-get install language-pack-zh-hans-base
apt install language-pack-gnome-zh-hans-base

# 中文输入法
apt-get install ibus-table-wubi ibus-pinyin ibus-sunpinyin
```

2.2.5.3 设置 gdm 自动登录

```
vim /etc/gdm3/custom.conf
# 修改下面的内容，ubuntu 即账户名
[daemon]
AutomaticLoginEnable=true
AutomaticLogin=ubuntu
TimedLoginEnable=true
TimedLogin=ubuntu
TimedLoginDelay=10
```

2.2.5.4 开机默认进图形界面

```
systemctl set-default graphical.target
```

2.2.6 rootfs 设置

2.2.6.1 设置 root 密码

```
passwd root
```

2.2.6.2 添加用户

```
useradd -s '/bin/bash' -m -G adm,sudo,video,plugdev,dialout ubuntu
```

2.2.6.3 设置时区

```
ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

2.2.6.4 设置 **hostname**

```
echo sytc > /etc/hostname
echo "127.0.0.1    sytc" >> /etc/hosts
```

2.2.6.5 设置 **bash** 为默认 **shell** (可选)

```
dpkg-reconfigure dash
```

选择 “NO”。

2.2.6.6 配置 **locale**

```
 dpkg-reconfigure locales
```

至少选择 en_US.UTF-8 和 zh_CN.UTF-8。

2.2.6.7 修改开机检测网络时间

修改开机检测网络时间，避免开机卡住:

```
vim /lib/systemd/system/networking.service
# 将里面的 TimeoutStartSec=5min 修改为
TimeoutStartSec=1sec
```

2.2.6.8 配置 fstab

配置/etc/fstab 内容如下：

```
# <file system>    <mount pt>   <type>   <options>        <dump>    <pass>
/dev/root          /            auto     rw,noauto         0          1
tmpfs              /tmp         tmpfs    mode=1777         0          0
tmpfs              /run         tmpfs    mode=0755,nosuid,nodev  0    0
#PARTLABEL=oem     /oem         ext2     defaults          0          2
PARTLABEL=userdata /userdata    ext4     defaults          0          0
proc               /proc        proc     defaults          0          0
devtmpfs           /dev         devtmpfs defaults          0          0
devpts  dev/pts    devpts   mode=0620,ptmxmode=0666,gid=5  0          0
tmpfs   /dev/shm   tmpfs        nosuid,nodev,noexec        0          0
sysfs              /sys         sysfs    defaults          0          0
debugfs            /sys/kernel/debug     debugfs defaults  0          0
pstore             /sys/fs/pstore        pstore  defaults  0          0
```

2.2.6.9 串口自动登录

修改文件 /usr/lib/systemd/system/serial-getty@.service 的 ExecStart 对应的值：

```
[Service]
# The '-o' option value tells agetty to replace 'login' arguments with an
# option to preserve environment (-p), followed by '--' for safety, and then
# the entered username.
ExecStart=-/sbin/agetty --autologin root --noclear %I $TERM
#ExecStart=-/sbin/agetty -o '-p -- \\u' --keep-baud 115200,38400,9600 %I $TERM
Type=idle
```

然后使能调试串口服务：

```
systemctl enable serial-getty@ttyFIQ0.service
```

2.2.6.10　设置 pulseaudio 音频服务 (可选)

添加文件 /etc/systemd/system/pulseaudio.service 内容如下:

```
[Unit]
Description=PulseAudio system server
# DO NOT ADD ConditionUser=!root

[Service]
Type=notify
ExecStart=pulseaudio --daemonize=no --system --realtime --log-target=journal
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

然后执行命令：

```
systemctl --system enable --now pulseaudio.service
```

添加以下内容到 /etc/pulse/client.conf 文件中：

```
default-server = /var/run/pulse/native
autospawn = no
```

将用户添加到 pulse-access 组中。

2.2.6.11　设置网卡 IP 地址

设置网络参数/etc/network/interfaces.d/eth0

```
auto eth0
    iface eth0 inet static
        address 192.168.1.80
```

```
        netmask 255.255.255.0
        gateway 192.168.1.1
```

修改/etc/network/interfaces *，执行命令：

```
echo "source-directory /etc/network/interfaces.d" > /etc/network/interfaces
```

用这里 7.2 的方法将网卡名固定为 ethX 。

2.2.6.12 首次开机扩容脚本

下载并安装附件的resize-disk 脚本。

```
tar xpvf resize-disk.tar.xz -C /
```

2.2.6.13 配置 ssh server

配置允许 root 登陆并解决登录慢的问题，将/etc/ssh/sshd_config 中的：

```
#PermitRootLogin prohibit-password
GSSAPIAuthentication yes
```

改为：

```
PermitRootLogin yes
GSSAPIAuthentication no
```

2.2.6.14 清理并退出

删除临时文件，卸载文件系统：

*这个步骤可选，默认的配置应该已经包含这个设置。

```
apt-get clean
exit
sudo umount ubuntu/dev/pts
sudo umount ubuntu/dev
sudo umount ubuntu/sys
sudo umount ubuntu/proc
```

2.3　打包成 ext4 文件系统

执行命令，打包成 ext4 的文件系统文件：

```
IMAGE_SIZE_MB=$(( $(sudo du -sh -m ubuntu | cut -f1) + 300 ))
dd if=/dev/zero of=ubuntu.img bs=1M count=0 seek=${IMAGE_SIZE_MB}
sudo mkfs.ext4 -d ubuntu ubuntu.img
```

可以将 ubuntu.img 烧写到 EMMC 的 rootfs 分区上。

3

制作欧拉文件系统

本章节介绍如何构建适用于 aarch64 的 openEuler 文件系统镜像。

3.1 准备安装环境

(1) 下载 rpi 的 openEuler 系统镜像文件

```
curl -O http://repo.openeuler.org/openEuler-24.03-LTS/raspi_img/\
    openEuler-24.03-LTS-raspi-aarch64.img.xz
xz -d openEuler-24.03-LTS-raspi-aarch64.img.xz
```

(2) 准备 aarch64 的系统：

```
sudo losetup -P /dev/loop0 openEuler-24.03-LTS-raspi-aarch64.img
sudo mount /dev/loop0p3 /media
mkdir rpi
sudo rsync -avzP /media/ rpi/
sudo umount /media
sudo losetup -d /dev/loop0
sudo cp /usr/bin/qemu-aarch64-static rpi/usr/bin
```

(3) 切换到安装环境

```
sudo mount --bind /dev rpi/dev
sudo mount --bind /dev/pts rpi/dev/pts
sudo mount -t proc proc rpi/proc
sudo mount -t sysfs sys rpi/sys
sudo chroot rpi /usr/bin/qemu-aarch64-static /bin/bash -i
```

(4) 编辑 /etc/resolv.conf

```
nameserver 8.8.8.8
nameserver 114.114.114.114
```

(5) 安装依赖包

```
dnf makecache
dnf install git wget make gcc bison dtc m4 flex bc openssl-devel \
    tar dosfstools rsync parted dnf-plugins-core tar
```

(6) 创建工作目录

```
export WORKDIR=/root/build
mkdir -p $WORKDIR
cd $WORKDIR
```

3.2　构建 rootfs 镜像

3.2.1 创建 RPM 数据库

```
cd $WORKDIR
mkdir -p rootfs/var/lib/rpm
rpm --root $WORKDIR/rootfs/ --initdb
```

3.2.2 下载安装 openEuler 发布包

```
rpm -ivh --nodeps --root $WORKDIR/rootfs/ \
    http://repo.openeuler.org/openEuler-24.03-LTS/everything/\
    aarch64/Packages/openEuler-release-24.03LTS-55.oe2403.aarch64.rpm
```

如出现错误：

“error: failed to exec scriptlet interpreter /bin/sh: No such file or directory”

可暂时忽略。

3.2.3 添加 yum 源

```
mkdir -p $WORKDIR/rootfs/etc/yum.repos.d
curl -o $WORKDIR/rootfs/etc/yum.repos.d/openEuler-24.03-LTS.repo \
     https://gitee.com/src-openeuler/openEuler-repos/raw/\
     openEuler-24.03-LTS/generic.repo
```

3.2.4 安装 dnf

```
dnf --installroot=$WORKDIR/rootfs/ install dnf --nogpgcheck -y
```

3.2.5 安装必要软件

```
dnf --installroot=$WORKDIR/rootfs/ makecache
dnf --installroot=$WORKDIR/rootfs/ install -y git wget bison dtc flex   \
    m4 dnf alsa-utils v4l-utils wpa_supplicant vim net-tools iproute    \
    iputils NetworkManager openssh-server passwd hostname lrzsz minicom \
    tcpdump ethtool openssl-devel tar dosfstools gdisk parted bc socat  \
    unzip gstreamer1 gstreamer1-plugins-base gstreamer1-plugins-good    \
    gstreamer1-plugins-bad-free gcc g++ gdb make cmake irqbalance sudo  \
    ntfs-3g rsync dnf-plugins-core pciutils usbutils
```

3.2.6 添加配置文件

3.2.6.1 设置 DNS

```
cp /etc/resolv.conf ${WORKDIR}/rootfs/etc/resolv.conf
```

其内容为：

```
nameserver 8.8.8.8
nameserver 114.114.114.114
```

3.2.6.2 设置 fstab

```
vim ${WORKDIR}/rootfs/etc/fstab
```

添加内容：

```
# <file system>     <mount pt>    <type>    <options>         <dump>    <pass>
/dev/root           /             auto      rw,noauto         0         1
tmpfs               /tmp          tmpfs     mode=1777         0         0
tmpfs               /run          tmpfs     mode=0755,nosuid,nodev  0   0
PARTLABEL=oem       /oem          ext2      defaults          0         2
PARTLABEL=userdata  /userdata     ext4      defaults          0         0
proc                /proc         proc      defaults          0         0
devtmpfs            /dev          devtmpfs  defaults          0         0
devpts   dev/pts    devpts    mode=0620,ptmxmode=0666,gid=5   0         0
tmpfs    /dev/shm   tmpfs         nosuid,nodev,noexec         0         0
sysfs               /sys          sysfs     defaults          0         0
debugfs             /sys/kernel/debug       debugfs defaults  0         0
pstore              /sys/fs/pstore          pstore  defaults  0         0
```

3.2.6.3 设置 IP 地址

```
mkdir $WORKDIR/rootfs/etc/sysconfig/network-scripts
vim $WORKDIR/rootfs/etc/sysconfig/network-scripts/ifcfg-eth0
```

内容如下：

```
NAME=eth0
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
IPADDR=192.168.1.80
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
```

3.2.6.4 禁用网卡名修改

```
vim $WORKDIR/rootfs/etc/udev/rules.d/80-net-setup-link.rules
```

内容如下：

```
# do not edit this file, it will be overwritten on update
SUBSYSTEM!="net", GOTO="net_setup_link_end"
IMPORT{builtin}="path_id"
ACTION=="remove", GOTO="net_setup_link_end"
IMPORT{builtin}="net_setup_link"
NAME=="", ENV{ID_NET_NAME}!="", NAME="$env{ID_NET_SLOT}"
LABEL="net_setup_link_end"
```

3.2.6.5 添加第一次开机脚本

在 $WORKDIR/rootfs/etc/rc.d/init.d/first-run.sh 写入以下内容:

```
#!/bin/bash
# chkconfig: - 99 10
# description: expand rootfs

ROOT_PART="$(findmnt / -o source -n)"  # /dev/mmcblk0p7
ROOT_DEV="/dev/$(lsblk -no pkname "$ROOT_PART")"  # /dev/mmcblk0
PART_NUM="$(echo "$ROOT_PART" | grep -o "[[:digit:]]*$")"  # 7

USER_PART="$(findmnt /userdata -o source -n)"  # /dev/mmcblk0p9
USER_NUM="$(echo "$USER_PART" | grep -o "[[:digit:]]*$")"  # 9

cat << EOF | gdisk $ROOT_DEV
p
w
Y
Y
EOF

parted -s $ROOT_DEV -- resizepart $PART_NUM 100%
resize2fs $ROOT_PART

parted -s $ROOT_DEV -- resizepart $USER_NUM 100%
```

```
resize2fs $USER_PART

depmod -a

if [ -f /etc/rc.d/init.d/first-run.sh ];
    then rm /etc/rc.d/init.d/first-run.sh;
fi
```

设置可执行权限：

```
chmod +x $WORKDIR/rootfs/etc/rc.d/init.d/first-run.sh
```

3.2.6.6 启用 rc.local 脚本

```
chmod +x $WORKDIR/rootfs/etc/rc.d/rc.local
```

3.2.6.7 安装 firmware 和 modules

从原文件系统中复制 firmware 和 modules:

```
cp -R /path/to/lib/firmware $WORKDIR/rootfs/usr/lib/
cp -R /path/to/lib/modules/5.10.110 $WORKDIR/rootfs/usr/lib/modules/
cp -R /path/to/usr/src/linux-5.10.110  $WORKDIR/rootfs/usr/src
```

3.2.6.8 串口自动登录

```
vim $WORKDIR/rootfs/usr/lib/systemd/system/serial-getty@.service
```

修改其中的 `ExecStart`：

```
[Service]
# The '-o' option value tells agetty to replace 'login' arguments with an
# option to preserve environment (-p), followed by '--' for safety, and then
# the entered username.
ExecStart=-/sbin/agetty --autologin root --noclear %I $TERM
#ExecStart=-/sbin/agetty -o '-p -- \\u' --keep-baud 115200,57600,38400,9600 - $TERM
Type=idle
```

3.3　rootfs 设置

3.3.1　挂载必要的路径

```
mount --bind /dev $WORKDIR/rootfs/dev
mount --bind /dev/pts $WORKDIR/rootfs/dev/pts
mount -t proc proc $WORKDIR/rootfs/proc
mount -t sysfs sys $WORKDIR/rootfs/sys
```

3.3.2　chroot 到 root

```
chroot $WORKDIR/rootfs /bin/bash
```

3.3.3　设置 root 密码

```
passwd
```

3.3.4　添加一个新用户

```
adduser euler
passwd euler
```

3.3.5 设置主机名

```
echo openEuler > /etc/hostname
echo "127.0.0.1 openEuler" >> /etc/hosts
```

3.3.6 设置默认时区为东八区

```
ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
```

3.3.7 安装 xfce4 桌面系统 (可选)

(1) 安装软件

```
dnf install -y dejavu-fonts liberation-fonts gnu-*-fonts wqy-zenhei-fonts
dnf install -y xorg-*
dnf install -y xfwm4 xfdesktop xfce4-* xfce4-*-plugin
dnf install -y lightdm lightdm-gtk
```

(2) 设置默认桌面为 XFCE 启动

```
echo 'user-session=xfce' >> /etc/lightdm/lightdm.conf.d/\
    60-lightdm-gtk-greeter.conf
echo 'autologin-user=euler' >> /etc/lightdm/lightdm.conf.d/\
    60-lightdm-gtk-greeter.conf
```

(3) 设置开机自启动图形界面

```
systemctl enable lightdm
systemctl set-default graphical.target
```

(4) 添加 /etc/X11/xorg.conf.d/20-modesetting.conf 文件，内容如下：

```
Section "Device"
    Identifier  "Rockchip Graphics"
    Driver      "modesetting"
```

```
### Use Rockchip RGA 2D HW accel
#    Option      "AccelMethod"    "exa"

### Use GPU HW accel
    Option      "AccelMethod"    "glamor"
    Option      "DRI"            "2"

### Set to "always" to avoid tearing, could lead to up 50% performance loss
    Option      "FlipFB"         "always"

### Limit flip rate and drop frames for "FlipFB" to reduce performance lost
#    Option      "MaxFlipRate"    "60"
    Option      "NoEDID"         "true"
    Option      "UseGammaLUT"    "true"

### Set virtual screen size (scaled by VOP hardware)
#    Option "VirtualSize" "DSI-1:600x1080"

### Set physical display paddings <top,bottom,left,right>
#    Option "Padding" "DSI-1:180,300,300,540"
EndSection

Section "Screen"
    Identifier  "Default Screen"
    Device      "Rockchip Graphics"
    Monitor     "Default Monitor"
EndSection

### Valid values for rotation are "normal", "left", "right"
Section "Monitor"
    Identifier  "Default Monitor"
    Option      "Rotate" "normal"
EndSection
```

3.3.8 安装 ukui 桌面系统 (可选)

ukui 是麒麟软件团队历经多年打造的一款 Linux 桌面，主要基于 GTK 和 QT 开发。与其他 UI 界面相比，ukui 更加注重易用性和敏捷度，各元件相依性小，可以不依赖其他套件而独自运行，给用户带来亲切和高效的使用体验。

(1) 安装 ukui

```
dnf install -y dejavu-fonts liberation-fonts gnu-*-fonts wqy-zenhei-fonts
dnf install ukui
```

(2) 配置允许 root 登陆

```
echo "greeter-show-manual-login=true" >> /usr/share/lightdm/lightdm.conf.d/\
    95-ukui-greeter.conf
echo "all-guest=false" >> /usr/share/lightdm/lightdm.conf.d/\
    95-ukui-greeter.conf
```

(3) 配置 euler 用户自动登陆

```
echo "autologin-user=euler" >> /usr/share/lightdm/lightdm.conf.d/\
    95-ukui-greeter.conf
```

(4) 添加 /etc/X11/xorg.conf.d/20-modesetting.conf 文件，内容如下：

```
Section "Device"
    Identifier  "Rockchip Graphics"
    Driver      "modesetting"

### Use Rockchip RGA 2D HW accel
#    Option      "AccelMethod"    "exa"

### Use GPU HW accel
    Option      "AccelMethod"    "glamor"
    Option      "DRI"            "2"

### Set to "always" to avoid tearing, could lead to up 50% performance loss
    Option      "FlipFB"         "always"

### Limit flip rate and drop frames for "FlipFB" to reduce performance lost
#    Option      "MaxFlipRate"    "60"
    Option      "NoEDID"         "true"
    Option      "UseGammaLUT"    "true"

### Set virtual screen size (scaled by VOP hardware)
#    Option "VirtualSize" "DSI-1:600x1080"

### Set physical display paddings <top,bottom,left,right>
#    Option "Padding" "DSI-1:180,300,300,540"
EndSection

Section "Screen"
```

```
    Identifier  "Default Screen"
    Device      "Rockchip Graphics"
    Monitor     "Default Monitor"
EndSection

### Valid values for rotation are "normal", "left", "right"
Section "Monitor"
    Identifier  "Default Monitor"
    Option      "Rotate" "normal"
EndSection
```

(5) 切换用户界面（GUI）模式

```
systemctl set-default graphical.target
```

还原到文本模式，用下面命令：

```
systemctl set-default multi-user.target
```

3.3.9 设置第一次开机脚本，然后退出

```
chkconfig --add first-run.sh
chkconfig first-run.sh on
dnf clean all
exit
```

3.3.10 取消临时挂载的目录

```
umount -l $WORKDIR/rootfs/dev/pts
umount -l $WORKDIR/rootfs/dev
umount -l $WORKDIR/rootfs/proc
umount -l $WORKDIR/rootfs/sys
```

3.4　制作镜像

(1) 计算镜像容量

```
IMAGE_SIZE_MB=$(( $(sudo du -sh -m $WORKDIR/rootfs | cut -f1) + 300 ))
```

(2) 创建镜像文件

```
dd if=/dev/zero of=rootfs.img bs=1M count=0 seek=${IMAGE_SIZE_MB}
```

(3) 格式化镜像

```
mkfs.ext4 -d $WORKDIR/rootfs rootfs.img
```

rootfs.img 就是最后可以用 rockchip 的工具烧写到 MMC 上的欧拉系统。

3.5　退出并清理

```
exit
sudo umount -l rpi/dev/pts
sudo umount -l rpi/dev
sudo umount -l rpi/proc
sudo umount -l rpi/sys
```

4

构建 openEuler Embedded

参考这里 https://embedded.pages.openeuler.org/master/introduction/index.html，把主要步骤记录一下。

4.1 系统安装 docker

用 root 权限执行下面的命令：

```
groupadd docker
usermod -a -G docker user
apt-get install docker.io
systemctl restart docker
chmod o+rw /var/run/docker.sock
```

4.2 准备 oebuild 安装环境

4.2.1 安装 oebuild

执行以下命令：

```
pip3 install oebuild
```

4.2.2 创建安装目录

执行以下命令：

```
oebuild init yocto-oebuild -b openEuler-24.03-LTS
cd yocto-oebuild
oebuild update
oebuild update docker
oebuild generate -p ok3588 -d rk3588 -f systemd
```

4.3　制作 Rk3588 镜像和 SDK

4.3.1 添加客户要求软件

编辑 “build/rk3588/compile.yaml”，在 “local.conf:” 下面添加必要软件，如下示例：

```
local_conf: |
  INIT_MANAGER = "systemd"
  VIRTUAL-RUNTIME_dev_manager = "systemd"
  IMAGE_INSTALL:append = "tcpdump ethtool iperf2 bridge-utils minicom"

  PREFERRED_PROVIDER_virtual/kernel ?= "linux-openeuler"
```

其中 “IMAGE_INSTALL:append” 是新增的内容。

4.3.2 编译镜像文件

```
cd build/rk3588
oebuild bitbake openeuler-image
oebuild bitbake openeuler-image -c do_populate_sdk
```

4.3.3 问题处理

(1) ERROR - ERROR: tcpdump-4.99.3-r0 do_fetch: Bitbake Fetcher Error: FetchError(‘Unable to fetch URL from any source.’, ‘file://tcpdump-4.99.3.tar.gz’)

先用下面命令下载文件：

```
cd yocto-oebuild/src/yocto-meta-openembedded/meta-networking/\
    recipes-support/tcpdump/tcpdump
wget -c http://www.tcpdump.org/release/tcpdump-4.99.3.tar.gz
wget -c https://gitee.com/src-openeuler/tcpdump/raw/master/\
    backport-0002-Use-getnameinfo-instead-of-gethostbyaddr.patch
wget -c https://gitee.com/src-openeuler/tcpdump/raw/master/\
    backport-0007-Introduce-nn-option.patch
wget -c https://gitee.com/src-openeuler/tcpdump/raw/master/\
    backport-0009-Change-n-flag-to-nn-in-TESTonce.patch
wget -c https://gitee.com/src-openeuler/tcpdump/raw/master/\
    tcpdump-Add-sw64-architecture.patch
```

然后修改“yocto-oebuild/src/yocto-meta-openeuler/meta-openeuler/dynamic-layers/networking-layer/recipes-support/tcpdump/tcpdump_%.bbappend” 文件中的 “SRC_URI[md5sum]” 和 “SRC_URI[sha256sum]”。例如：

```
SRC_URI[md5sum] = "491aeb15c1c72d59b9288a5a6953e8a9"
SRC_URI[sha256sum] = "ad75a6ed3dc0d9732945b2e5483cb41dc8b4b528a169315e499c6861952e73b3"
```

5

制作 ext2 格式的 boot.img

U-Boot 可以通过 “run boot_extlinux” 来启动 ext2 格式的镜像文件，这里介绍一下如何制作这种格式的 boot.img 文件：

```
mkdir -p tmpdir/extlinux
cp arch/arm64/boot/Image tmpdir
cp arch/arm64/boot/dts/rockchip/rk3588-sytc.dtb tmpdir
cat >> tmpdir/extlinux/extlinux.conf << EOF
label kernel-5.10.110
    kernel /Image
    fdt /rk3588-sytc.dtb
    append earlycon=uart8250,mmio32,0xfeb50000 console=ttyFIQ0 \
        irqchip.gicv3_pseudo_nmi=0 root=PARTLABEL=rootfs rw rootwait
EOF
dd if=/dev/zero of=boot.img bs=1M count=0 seek=64
mkfs.ext2 -d tmpdir boot.img
e2fsck -p -f boot.img
resize2fs -M boot.img
```

注意：U-Boot 需要环境变量 “distro_bootpart” 来指定 boot.img 所在的分区。如果没有正常加载，在 uboot 的命令行上查看它是否有正确的值。

插图目录