

Purple Pi OH2 Linux开发手册

1 上手指南

1.介绍

2 Linux开发

2.1 获取 SDK

2.2 校验md5值

2.3 解压

2.4 SDK目录介绍

2.5 配置文件介绍

2.6 分区说明

3.编译

3.1 选择项目

3.2 指定编译文件系统

3.3 自动编译

3.4 分步编译

3.3.1.编译uboot

3.3.2.编译kernel

3.3.3.编译buildroot系统

3.3.4.编译ubuntu镜像

3.3.5.编译debian镜像

3.3.6.编译recovery

3.3.7.固件打包

4 烧录说明

4.1.整包烧录

4.1.1.运行RK开发工具

4.1.2.点击升级固件

4.1.3.点击固件

4.1.4.点击升级

4.2.分包烧录

4.2.1.更新下载地址

4.2.2.烧录

5.驱动开发

5.1.CAN

5.1.1.CAN简介

5.1.2.DTS 节点配置

配置CAN

通信测试

5.2.Ethernet

公共配置

板级的配置

5.3.RTC

5.4.UART

DTS配置

收发测试



www.industio.cn

Purple Pi OH2

Linux开发手册

文档修订历史

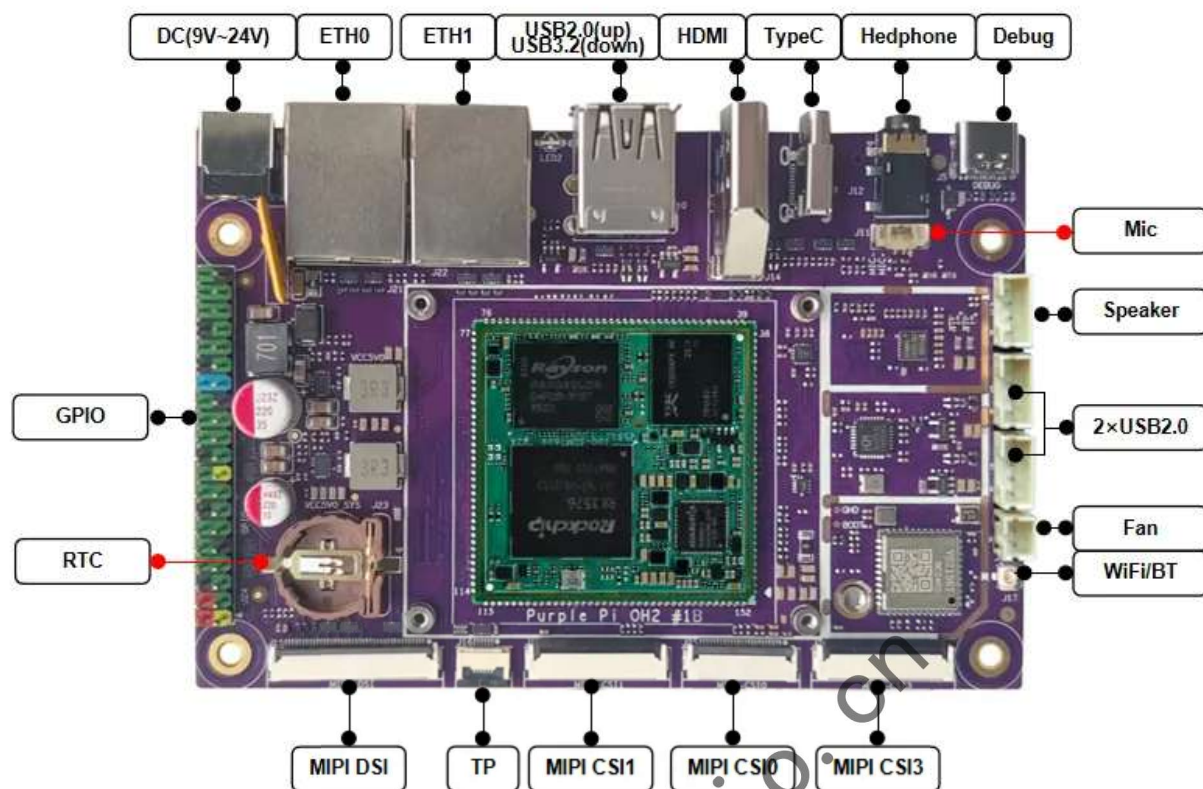
版本	PCBA版本	修订内容	修订	审核	日期
V1.0	V1B	创建文档	TWX		2025/09/04
V1.0	V1B	更正编译buildroot文件系统方法	TWX		2025/09/08

1 上手指南

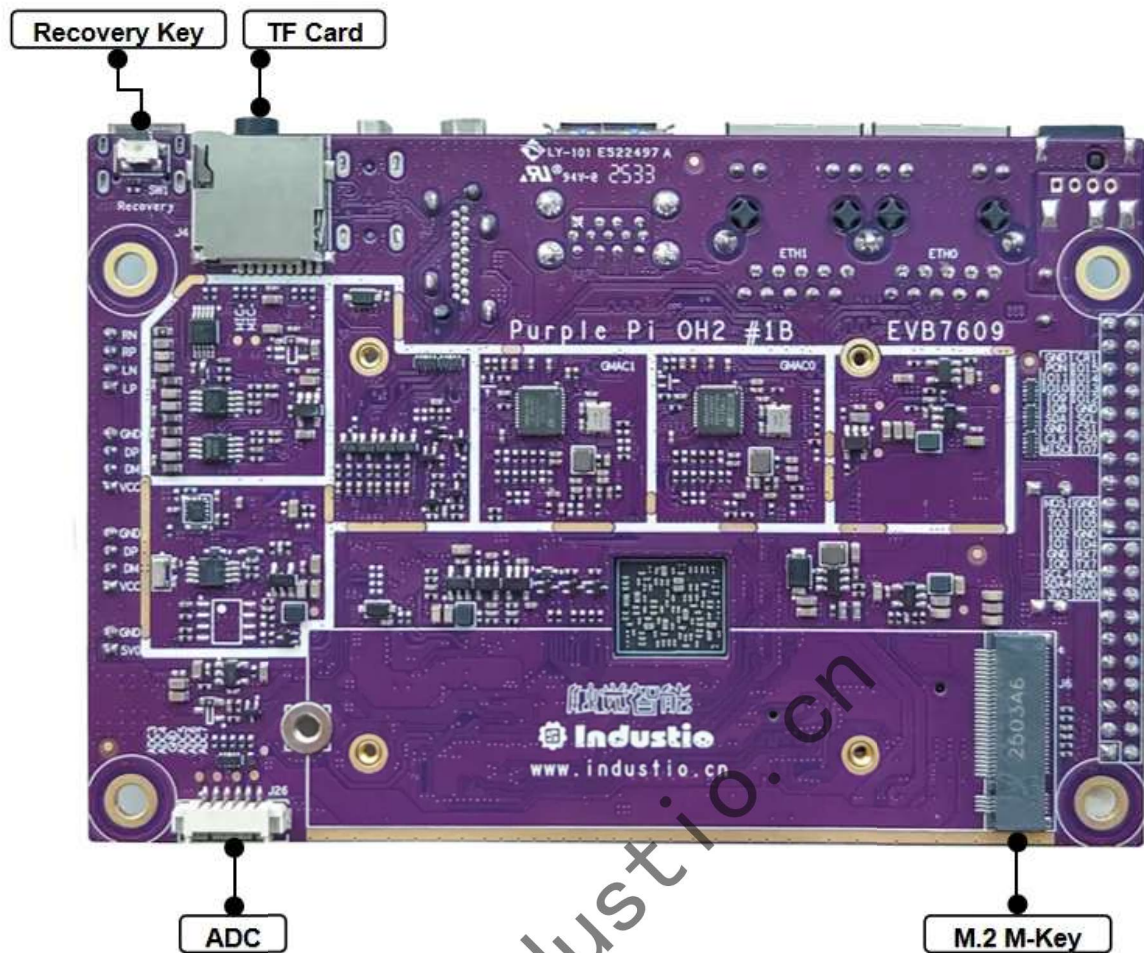
1.介绍

Purple Pi OH2 配置 Rockchip 八核 64 位 AIOT 处理器 RK3576，采用先进工艺制程，高性能低功耗，内置 ARM Mali G52 MC3 GPU，集成6 TOPS算力NPU，支持主流大模型的私有化部署。具备 4K@120fps 高清高帧率显示能力，支持外部看门狗，具有实时网络、Flexbus、硬件资源隔离、DSMC 等工业新特性，满足不同的应用需求。

Purple Pi OH2正面接口位号图，如下图所示：



Purple Pi OH2背面接口位号图，如下图所示：



2 Linux开发

2.1 获取 SDK

下载路径：百度网盘/Purple Pi OH2-Linux/4.软件资料/系统SDK/

2.2 校验md5值

```
Bash |  
1 $ md5sum rk3576_purple_pi_oh2_linux_sdk.tar.gz
```

2.3 解压

首先准备一个空文件夹用于存放 SDK，建议在 home 目录下，本文以~/purple_pi_oh2为例。

```
1  $ mkdir ~/purple_pi_oh2
2  $ cd ~/purple_pi_oh2
3  $ tar -xvf rk3576_purple_pi_oh2_linux_sdk.tar.gz -C ./
4
5  $ cd rk3576_purple_pi_oh2_linux_sdk
6  $ git checkout purple_pi_oh2
7
8  #查看分支(默认分支为purple_pi_oh2)
9  $ git branch
10 * purple_pi_oh2
11     purple_pi_oh2_preempt_rt
12     purple_pi_oh2_xenomai_rt
13
14 #获取代码
15 $ git reset --hard
```

2.4 SDK目录介绍

```
1  $ tree -L 1
2  .
3  ├── app
4  ├── buildroot
5  ├── build.sh -> device/rockchip/common/scripts/build.sh
6  ├── common -> device/rockchip/common
7  ├── Copyright_Statement.md -> docs/licenses/LICENSE
8  ├── debian
9  ├── debian_rootfs
10 ├── device
11 ├── docs
12 ├── external
13 ├── kernel -> kernel-6.1
14 ├── kernel-6.1
15 ├── Makefile -> device/rockchip/common/Makefile
16 ├── output
17 ├── prebuilts
18 ├── README.md -> device/rockchip/common/README.md
19 ├── rkbin
20 ├── rkflash.sh -> device/rockchip/common/scripts/rkflash.sh
21 ├── rockdev -> output/firmware
22 ├── tools
23 ├── u-boot
24 └── yocto
```

```
1  app: 存放上层应用 APP, 主要是一些应用Demo。
2  buildroot: 基于 Buildroot (2024) 开发的根文件系统。
3  device/rockchip: 存放芯片板级配置以及SDK编译和打包固件的脚本和文件等。
4  docs: 存放通用开发指导文档、芯片平台相关文档、Linux 系统开发相关文档、其他参考文档等。
5  external: 存放第三方相关仓库, 包括显示、音视频、摄像头、网络、安全等。
6  kernel: 存放 Kernel 开发的代码。
7  output: 存放每次生成的固件信息、编译信息、XML、主机环境等。
8  prebuilts: 存放交叉编译工具链。
9  rkbin: 存放 Rockchip 相关二进制和工具。
10 rockdev: 存放编译输出固件, 实际软链接到 output/firmware 。
11 tools: 存放 Linux 和 Window 操作系统下常用工具。
12 u-boot: 存放基于 v2017.09 版本进行开发的 U-Boot 代码。
13 yocto: 存放yocto相关编译脚本和代码。
```

注意:

1. SDK 采用交叉编译, 所以要在 X86_64 电脑上使用 SDK, 不要将 SDK 下载到板子上。

2. 编译环境请使用 Ubuntu22.04（真机或 虚拟机），如果使用其他版本可能导致编译出错。
3. 不要在虚拟机共享文件夹以及非英文目录存放、解压SDK。
4. 获取、编译 SDK 请全程使用普通用户，不允许也不需要使用 root 权限（除非需要 apt 安装软件）

2.5 配置文件介绍

在 `device/rockchip/rk3576` 目录下，有不同板型的配置文件(xxxx_defconfig)，用于管理 SDK 每个环节的编译配置。rockchip_rk3576_ido_evb7609_v1*_defconfig 作为基础的配置。其他版型的配置文件会引用 rockchip_rk3576_ido_evb7609_v1*_defconfig 并且覆盖其中的一些变量配置。以 rockchip_rk3576_ido_evb7609_v1_hdmi_defconfig 做介绍：

```
1 RK_BUILDROOT_BASE_CFG="rk3576"
2 RK_CHIP_FAMILY="rk3576"           #芯片类型
3 # RK_WIFIBT is not set
4 RK_UBOOT_SPL=y
5 RK_KERNEL_DTS_NAME="ido-evb7609-v1a-hdmi" # 板型 dts 名称
6 RK_USE_FIT_IMG=y
7 RK_KERNEL_CFG="rockchip_linux_rk3576_defconfig" # kernel 编译配置
```

2.6 分区说明

parameter 文件中包含了固件的分区信息，以 parameter.txt 为例：

```

1  FIRMWARE_VER: 1.0
2  MACHINE_MODEL: RK3576
3  MACHINE_ID: 007
4  MANUFACTURER: RK3576
5  MAGIC: 0x5041524B
6  ATAG: 0x00200800
7  MACHINE: 0xffffffff
8  CHECK_MASK: 0x80
9  PWR_HLD: 0,0,A,0,1
10 TYPE: GPT
11 GROW_ALIGN: 0
12 CMDLINE: mtdparts=:0x00002000@0x00004000(uboot),0x00002000@0x00006000(misc),0x00020000@0x00008000(boot),0x00040000@0x00028000(recovery),0x00010000@0x00068000(backup),0x00800000@0x00078000(userdata),0x00040000@0x00878000(overlay),-@0x008B8000(rootfs:grow)
13 uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9
14 uuid:boot=7A3F0000-0000-446A-8000-702F00006273

```

CMDLINE 属性是我们关注的地方，以 uboot 为例，0x00002000@0x00004000(uboot) 中 0x00002000 为 uboot 分区的起始位置，0x00004000 为分区的大小，以此类推。

3. 编译

3.1 选择项目

```

1  xxx$ ./build.sh lunch
2  1. rockchip_rk3576_ido_evb7609_v1_hdmi_defconfig
3  2. rockchip_rk3576_ido_evb7609_v1_mipi_1920x1200_defconfig
4  Which would you like? [1]: 2
5  ...

```

选择对应的数字即可。

3.2 指定编译文件系统

设置环境变量 RK_ROOTFS_SYSTEM 为 buildroot/debian/ubuntu，即可指定编译的文件系统。

如需要编译buildroot文件系统：

```
▼ Bash |
1 export RK_R00TFS_SYSTEM=buildroot
```

如需要编译debian文件系统：

```
▼ Bash |
1 export RK_R00TFS_SYSTEM=debian
```

如需要编译ubuntu文件系统：

```
▼ Bash |
1 export RK_R00TFS_SYSTEM=ubuntu
```

3.3 自动编译

```
▼ Bash |
1 $ ./build.sh
```

固件编译后生成路径：rockdev/update.img

3.4 分步编译

3.3.1.编译uboot

```
▼ Bash |
1 ./build.sh uboot
```

固件编译后生成路径：rockdev/uboot.img

3.3.2.编译kernel

▼

Bash |

```
1 ./build.sh kernel
```

固件编译后生成路径：rockdev/boot.img

3.3.3.编译buildroot系统

▼

Bash |

```
1 ./build.sh buildroot
```

固件编译后生成路径：rockdev/rootfs.img

3.3.4.编译ubuntu镜像

▼

Bash |

```
1 ./build.sh ubuntu
```

3.3.5.编译debian镜像

▼

Bash |

```
1 ./build.sh debian
```

3.3.6.编译recovery

▼

Bash |

```
1 ./build.sh recovery
```

固件编译后生成路径：rockdev/recovery.img

3.3.7.固件打包

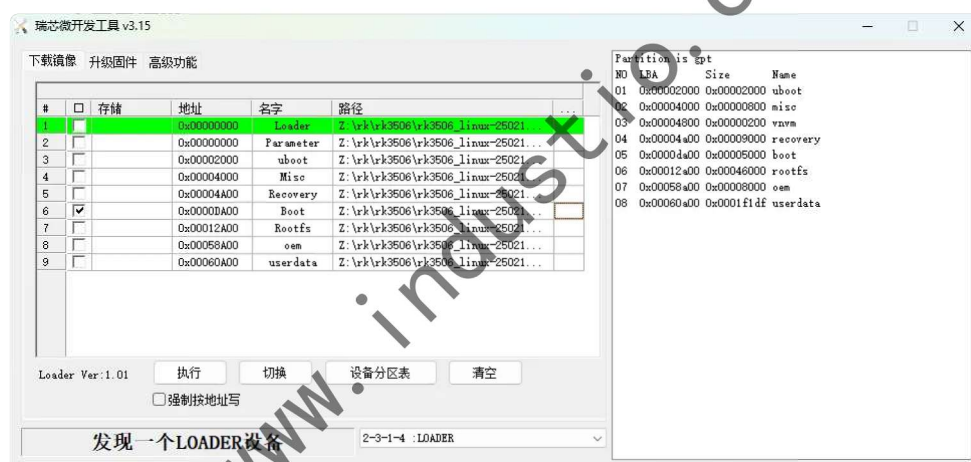
```
1 ./build.sh updateimg
```

4 烧录说明

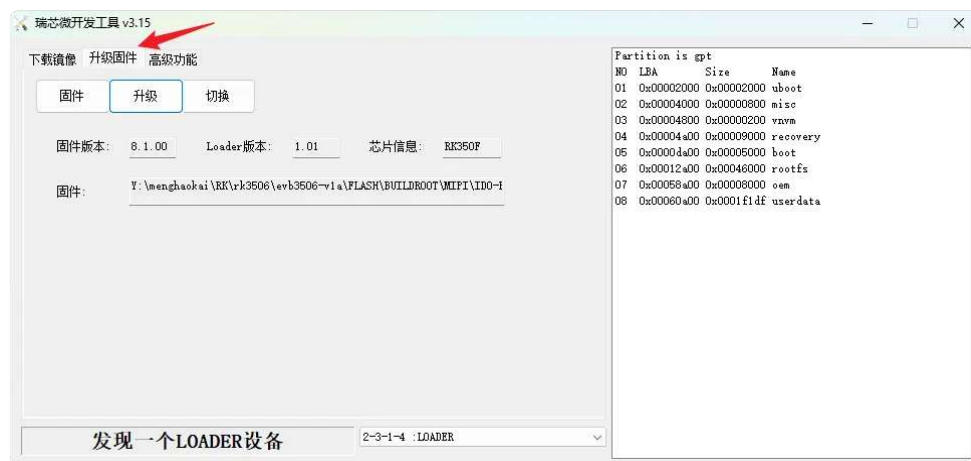
注意：板子需要进入到烧录模式，具体方法可以参考：《Purple Pi OH2 开发板固件烧录手册》

4.1.整包烧录

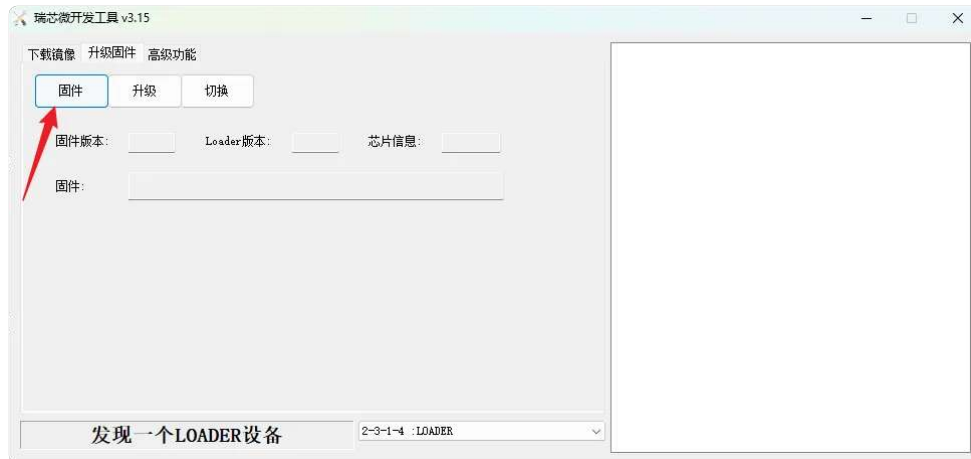
4.1.1.运行RK开发工具



4.1.2.点击升级固件



4.1.3.点击固件



选择: `rockdev/update.img`

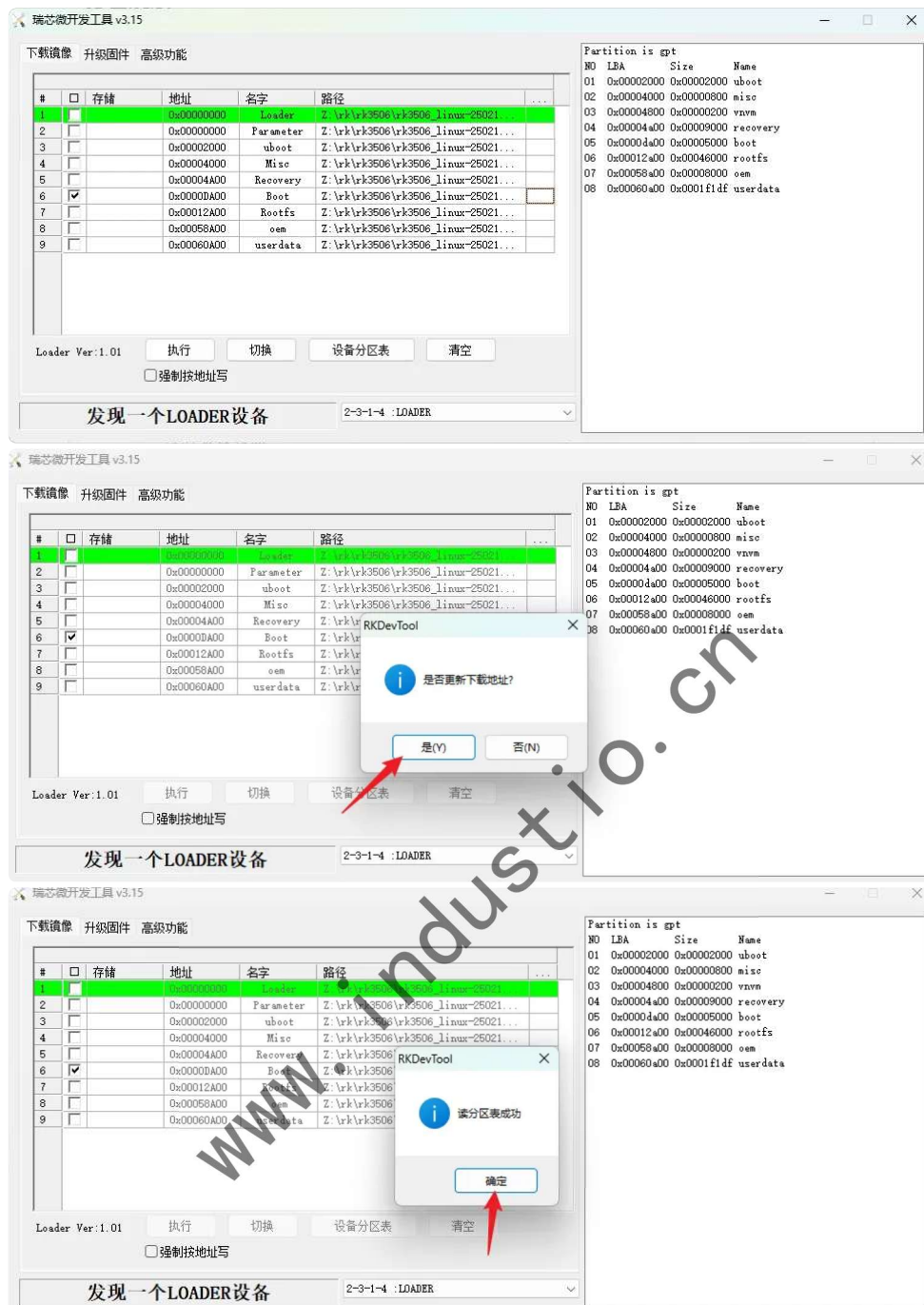
4.1.4.点击升级



点击升级后，右侧会开始下载固件；下载完成后，板子会自动启动。

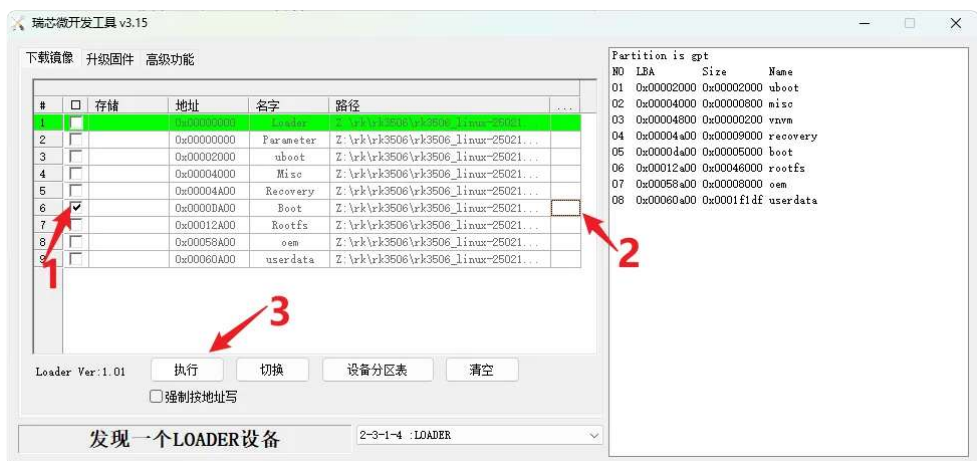
4.2.分包烧录

4.2.1.更新下载地址



4.2.2.烧录

1. 在对应的分区前选择对号
2. 然后点击右侧去选择对应的镜像
3. 点击执行即可烧录



5.驱动开发

5.1.CAN

5.1.1.CAN简介

CAN(Controller Area Network)总线，即控制器局域网总线，是一种有效支持分布式控制或实时控制的串行通信网络。CAN总线是一种在汽车上广泛采用的总线协议，被设计作为汽车环境中的微控制器通讯。

5.1.2.DTS 节点配置

公共配置：kernel/arch/arm64/boot/dts/rk3576.dtsi

```
1 can0: can@2ac00000 {
2     compatible = "rockchip,rk3576-canfd";
3     reg = <0x0 0x2ac00000 0x0 0x1000>;
4     interrupts = <GIC_SPI 121 IRQ_TYPE_LEVEL_HIGH>;
5     clocks = <&cru CLK_CAN0>, <&cru HCLK_CAN0>;
6     clock-names = "baudclk", "apb_pclk";
7     resets = <&cru SRST_CAN0>, <&cru SRST_H_CAN0>;
8     reset-names = "can", "can-apb";
9     dmas = <&dmac0 20>;
10    dma-names = "rx";
11    status = "disabled";
12 };
13
14 can1: can@2ac10000 {
15     compatible = "rockchip,rk3576-canfd";
16     reg = <0x0 0x2ac10000 0x0 0x1000>;
17     interrupts = <GIC_SPI 122 IRQ_TYPE_LEVEL_HIGH>;
18     clocks = <&cru CLK_CAN1>, <&cru HCLK_CAN1>;
19     clock-names = "baudclk", "apb_pclk";
20     resets = <&cru SRST_CAN1>, <&cru SRST_H_CAN1>;
21     reset-names = "can", "can-apb";
22     dmas = <&dmac1 21>;
23     dma-names = "rx";
24     status = "disabled";
25 };
26
```

配置CAN

板级配置: ido-evb7609-v1a-extend.dtsi

```
1 &can0 {
2     assigned-clocks = <&cru CLK_CAN0>;
3     assigned-clock-rates = <200000000>;
4     pinctrl-names = "default";
5     pinctrl-0 = <&can0m2_pins>;
6     status = "okay";
7 };
8
9 &can1 {
10    assigned-clocks = <&cru CLK_CAN1>;
11    assigned-clock-rates = <200000000>;
12    pinctrl-names = "default";
13    pinctrl-0 = <&can1m3_pins>;
14    status = "okay";
15 };
16
```

通信测试

```
1 #关闭can0设备
2 ip link set can0 down
3
4 #普通can协议 (1.0)
5 ip link set can0 type can bitrate 1000000 dbitrate 1000000 fd on
6
7 #查看can信息, 波特率等等
8 ip -details link show can0
9
10 #启动can0
11 ip link set can0 up
12
13 #执行candump, 阻塞等待can0接收
14 candump can0
15
16 #can格式发送
17 cansend can0 123#DEADBEEF12345679
```

5.2.Ethernet

公共配置

```
kernel/arch/arm64/boot/dts/rk3576.dtsi
```

www.industio.cn

```

1  gmac0: ethernet@2a220000 {
2      compatible = "rockchip,rk3576-gmac", "snps,dwmac-4.20a";
3      reg = <0x0 0x2a220000 0x0 0x10000>;
4      interrupts = <GIC_SPI 293 IRQ_TYPE_LEVEL_HIGH>,
5                  <GIC_SPI 298 IRQ_TYPE_LEVEL_HIGH>;
6      interrupt-names = "macirq", "eth_wake_irq";
7      rockchip,grf = <&sdgmac_grf>;
8      rockchip,php_grf = <&ioc_grf>;
9      clocks = <&cru CLK_GMAC0_125M_SRC>, <&cru CLK_GMAC0_RMII_CRU>,
10             <&cru PCLK_GMAC0>, <&cru ACLK_GMAC0>,
11             <&cru CLK_GMAC0_PTP_REF>;
12      clock-names = "stmmaceth", "clk_mac_ref",
13                   "pclk_mac", "aclk_mac",
14                   "ptp_ref";
15      resets = <&cru SRST_A_GMAC0>;
16      reset-names = "stmmaceth";
17      power-domains = <&power RK3576_PD_SDGMAC>;
18
19      dma-coherent;
20      snps,mixed-burst;
21      snps,tso;
22
23      snps,axi-config = <&gmac0_stmmac_axi_setup>;
24      snps,mtl-rx-config = <&gmac0_mtl_rx_setup>;
25      snps,mtl-tx-config = <&gmac0_mtl_tx_setup>;
26      status = "disabled";
27
28  mdio0: mdio {
29      compatible = "snps,dwmac-mdio";
30      #address-cells = <0x1>;
31      #size-cells = <0x0>;
32  };
33
34  gmac0_stmmac_axi_setup: stmmac-axi-config {
35      snps,wr_osr_lmt = <4>;
36      snps,rd_osr_lmt = <8>;
37      snps,blen = <0 0 0 0 16 8 4>;
38  };
39
40  gmac0_mtl_rx_setup: rx-queues-config {
41      snps,rx-queues-to-use = <1>;
42      queue0 {};
43  };
44

```

```

45     gmac0_mtl_tx_setup: tx-queues-config {
46         snps,tx-queues-to-use = <1>;
47         queue0 {};
48     };
49 };

```

板级的配置

```

1  &gmac0 {
2      /* Use rgmii-rxid mode to disable rx delay inside Soc */
3      phy-mode = "rgmii";
4      clock_in_out = "output";
5
6      snps,reset-gpio = <&gpio0 RK_PC7 GPIO_ACTIVE_LOW>;
7      snps,reset-active-low;
8      /* Reset time is 20ms, 100ms for rtl8211f */
9      snps,reset-delays-us = <0 20000 100000>;
10
11     pinctrl-names = "default";
12     pinctrl-0 = <&eth0m0_miim
13                 &eth0m0_tx_bus2
14                 &eth0m0_rx_bus2
15                 &eth0m0_rgmii_clk
16                 &eth0m0_rgmii_bus >;
17     //&ethm0_clk0_25m_out>;
18
19     tx_delay = <0x1a>;
20     rx_delay = <0x26>;
21
22     phy-handle = <&rgmii_phy0>;
23     status = "okay";
24 };

```

5.3.RTC

Purple Pi OH2 采用 HYM8563 作为RTC(*Real Time Clock*) , 需要接入 RTC 电池给 RTC 芯片供电可以保证在短时间系统断电后 RTC 能正常运行。

DTS配置参考: `kernel/arch/arm64/boot/dts/ido-evb7609_v1b.dtsi`

驱动参考: `kernel/drivers/rtc/rtc-hym8563.c`

Linux下的rtc使用方法为:

```
1  #同步网络时间
2  $ ntpdate cn.pool.ntp.org
3
4  #查看当前 RTC 的日期和时间:
5  $ cat /sys/class/rtc/rtc0/date
6  2025-05-27
7
8  $ cat /sys/class/rtc/rtc0/time
9  10:10:30
10
11 $ 查看系统时间
12 $ date
13 Tue May 27 18:10:41 CST 2025
14
15 #设置系统时间
16 $ date -s "2024-10-09 14:02:30"
17
18 #将rtc时间调整为与目前的系统时间一致
19 $ hwclock -w
20
21 #获取硬件rtc当前时间（断电重启读取时间没有太大偏差）
22 $ hwclock -r
23 2024-10-09 14:02:35.945604+00:00
24
25 #将rtc时间设置为系统时间
26 hwclock --systohc
```

5.4.UART

DTS配置

```
1 &uart3 {
2     status = "okay";
3     pinctrl-names = "default";
4     pinctrl-0 = <&uart3m0_xfer>;
5 };
```

配置好串口后，硬件接口对应软件上的节点为：

```
▼ Bash |
1  UART3:  /dev/ttyS3
```

收发测试

使用microcom可以进行串口收发测试，命令如下：

```
▼ Bash |
1  root@rk3576:~# microcom -s 115200 -p /dev/ttyS3
```

注意：测试完成，按Ctrl+x退出。

www.industio.cn