IDO-EVB3506-V1 Linux开发手册

IDO-EVB3506-V1

Linux开发手册

深圳触觉智能科技有限公司

www.industio.cn

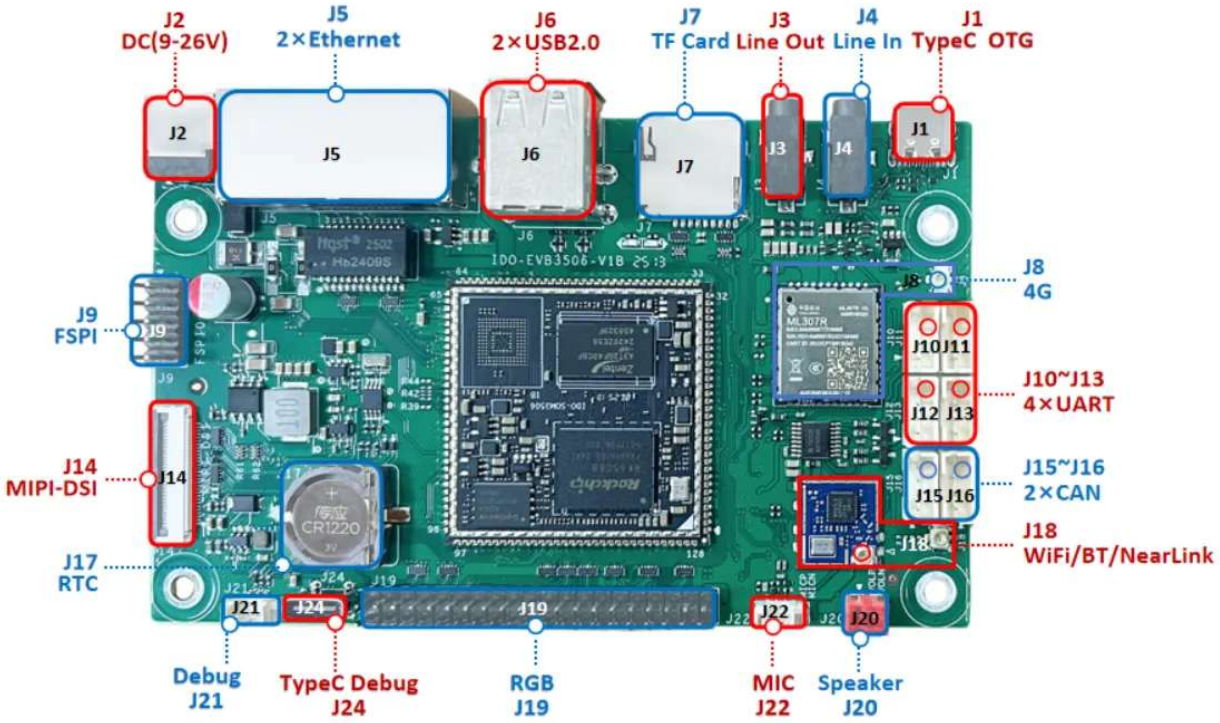
文档修订历史

版本	PCBA版本	修订内容	修订	审核	日期
V1.0	V1B	创建文档	MHK	IDO	2025/04/01
V1.1	V1B	优化文档	WJY	IDO	2025/04/10

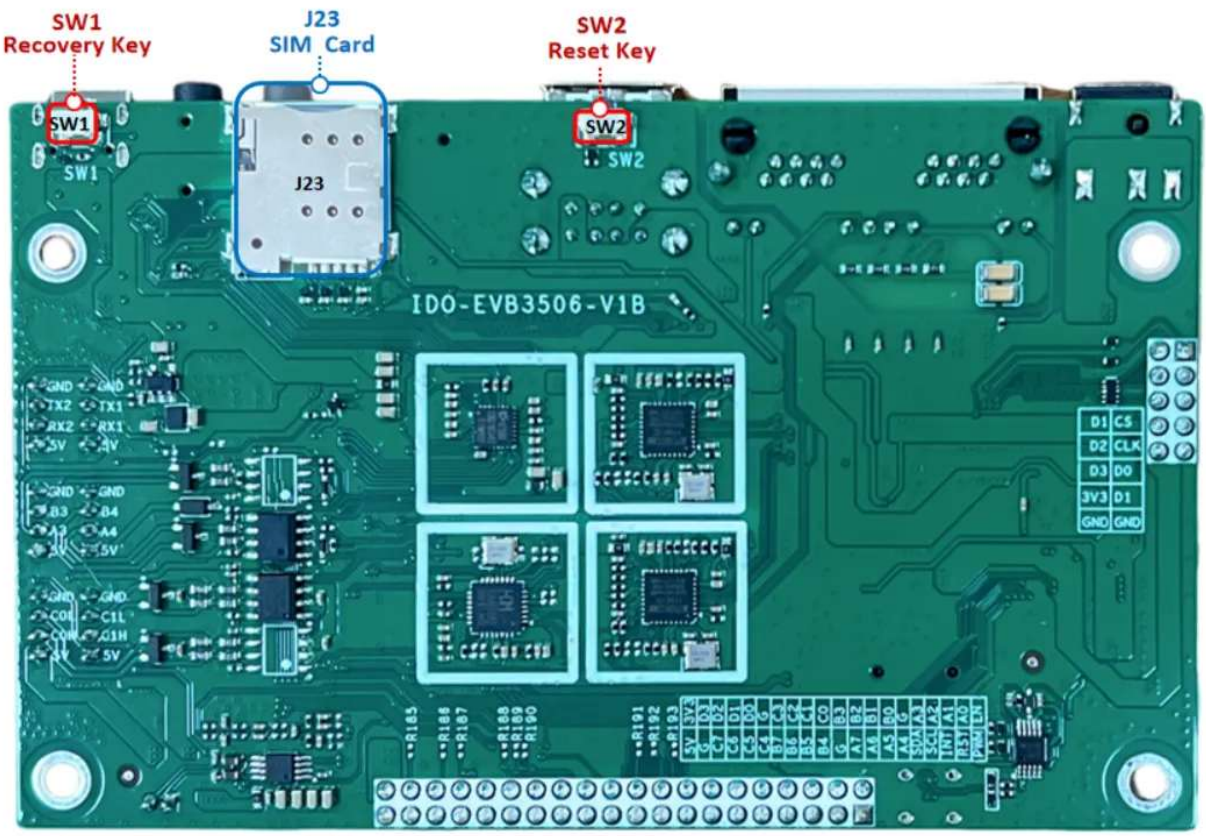
1 上手指南

1.介绍

IDO-EVB3506是搭载 Rockchip 全新芯片 RK3506B，22nm 先进制程工艺，集成了三核 ARM Cortex-A7和 Cortex-M0，主频高达 1.5GHz。支持 AMP 多核异构架构，一颗芯片可支持 Linux、RTOS、Bare-metal 灵活组合搭配，采用标准 RPMsg 核间通信机制。SDK 原生支持 LVGL 轻量级 UI 框架，内部2D硬件加速让 LVGL 运行更加流畅。



IDO-EVB3506-V1正面



IDO-EVB3506-V1背面

2 Linux开发

1.获取 SDK

下载路径：百度网盘/Linux/软件资料/SDK/

2.校验md5值

```Bash
$ md5sum rk3506_linux6.1_rkr4_v1.tar.gz
```

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~/evb3506为例

```Bash
$ mkdir ~/evb3506
$ cd ~/evb3506
$ tar -xvf rk3506_linux6.1_rkr4_v1.tar.gz -C ./
```

4.SDK目录介绍

```Bash
.
├── app
├── buildroot
├── build.sh -> device/rockchip/common/scripts/build.sh
├── check_git.sh
├── device
├── docs
├── envsetup.sh -> buildroot/scripts/envsetup.sh
├── external
├── hal
├── kernel -> kernel-6.1
├── kernel-6.1
├── Makefile -> device/rockchip/common/Makefile
├── prebuilts
├── rkbin
├── rkflash.sh -> device/rockchip/common/scripts/rkflash.sh
├── rootfs
├── rtos
├── tools
├── u-boot
└── yocto
```

```Bash
app：存放上层应用 APP，主要是一些应用Demo。
buildroot：基于 Buildroot（2024）开发的根文件系统。
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。
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。
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。
kernel：存放 Kernel 开发的代码。
output：存放每次生成的固件信息、编译信息、XML、主机环境等。
prebuilts：存放交叉编译工具链。
rkbin：存放 Rockchip 相关二进制和工具。
rockdev：存放编译输出固件,实际软链接到 output/firmware 。
tools：存放 Linux 和 Window 操作系统下常用工具。
u-boot：存放基于 v2017.09 版本进行开发的 U-Boot 代码。
rtos：存放rtos相关源码。
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）

5.编译

1.选择项目

```Bash
mhkwork@dell-PowerEdge-R430:~/work/rk/rk3506/release/rk3506_linux6.1_rkr4_v1$ ./build.sh lunch
mhkwork@dell-PowerEdge-R430:~/work/rk/rk3506/rk3506_linux-250211/rk3506_linux6.1$ ./build.sh lunch
Log colors: message notice warning error fatal

Log saved at /mnt/5c953a98-9ee6-45b9-8cea-a2898eb313d7/mhk/rk/rk3506/rk3506_linux-250211/rk3506_linux6.1/output/sessions/2025-04-11_15-01-58
Pick a defconfig:

1. ido-evb3506-v1a-mipi-720x720-emmc_defconfig
2. ido-evb3506-v1a-mipi-720x720-nand-rt_defconfig
3. ido-evb3506-v1a-mipi-720x720-nand_defconfig
4. ido-evb3506-v1a-rgb-1024x600-emmc_defconfig
5. ido-evb3506-v1a-rgb-1024x600-nand_defconfig
6. rockchip_rk3502_robot_defconfig
7. rockchip_rk3506_b_evb1_defconfig
8. rockchip_rk3506_g_demo_defconfig
9. rockchip_rk3506_g_evb1_amp_defconfig
10. rockchip_rk3506_g_evb1_defconfig
11. rockchip_rk3506_g_evb1_smp_defconfig
Which would you like? [1]:

```

根据硬件版本和屏幕配置选择对应的工程即可。

2.自动编译

```Bash
$ ./build.sh
```

固件编译后生成路径：rockdev/update.img

3.分步编译

1.编译uboot

```bash
./build.sh uboot
```

固件编译后生成路径：rockdev/uboot.img

2.编译kernel

```bash
./build.sh kernel
```

固件编译后生成路径：rockdev/boot.img

3.编译buildroot系统

```bash
./build.sh buildroot
```

固件编译后生成路径：rockdev/rootfs.img

4.编译ubuntu镜像

```bash
./build.sh ubuntu
```

注意：nand配置不支持ubuntu系统。

5.编译recovery

```bash
./build.sh recovery
```

固件编译后生成路径：rockdev/recovery.img

6.固件打包

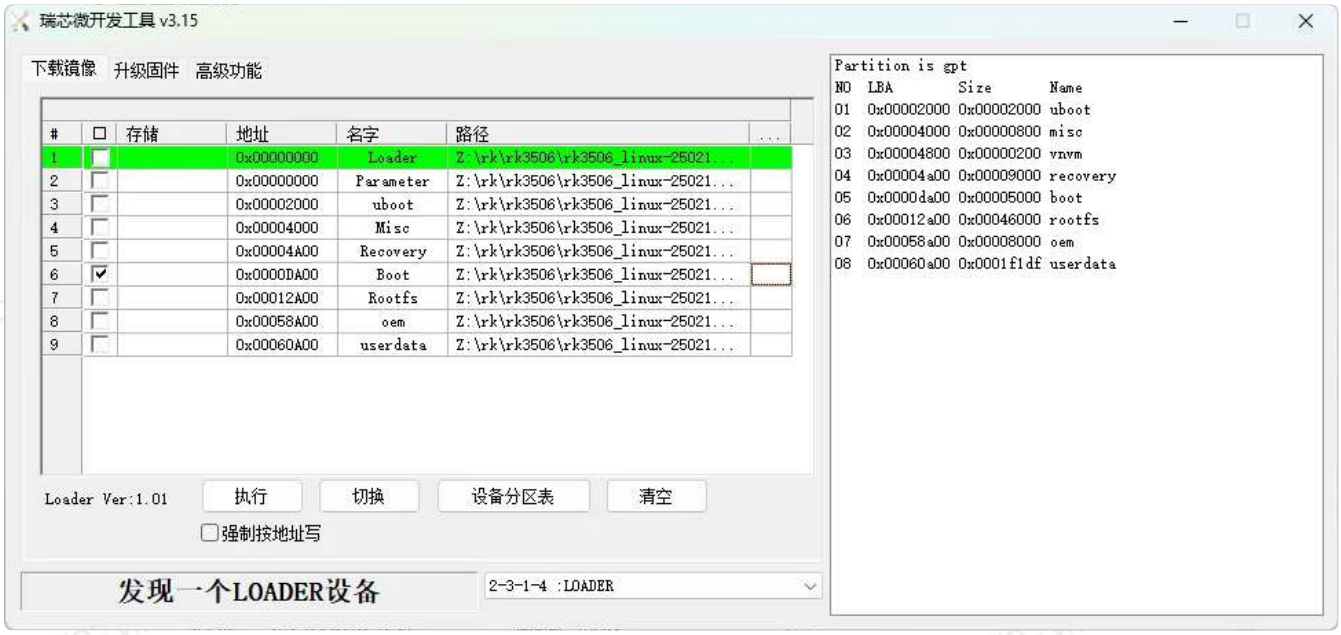
```bash
./build.sh updateimg
```

6烧录说明

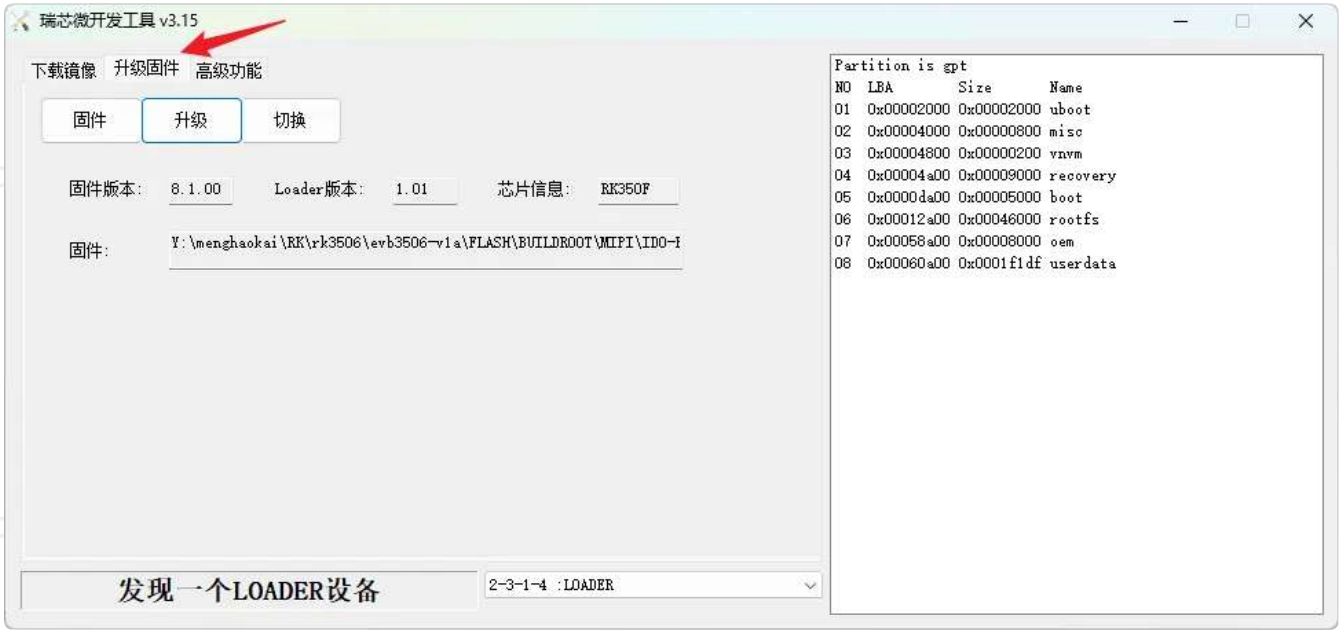
注意：板子需要进入到烧录模式，具体方法可以参考：《IDO-EVB3506-V1开发板固件烧录手册》

1.整包烧录

1.运行RK开发工具



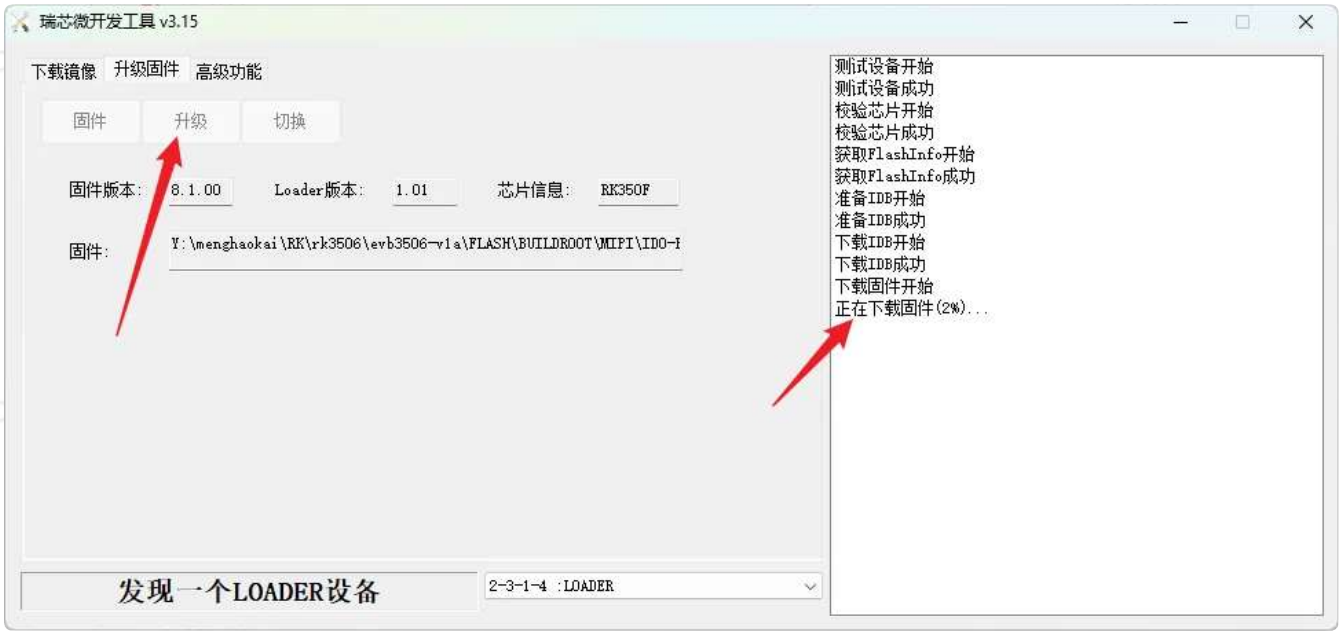
2.点击升级固件



3.点击固件

选择：rockdev/update.img

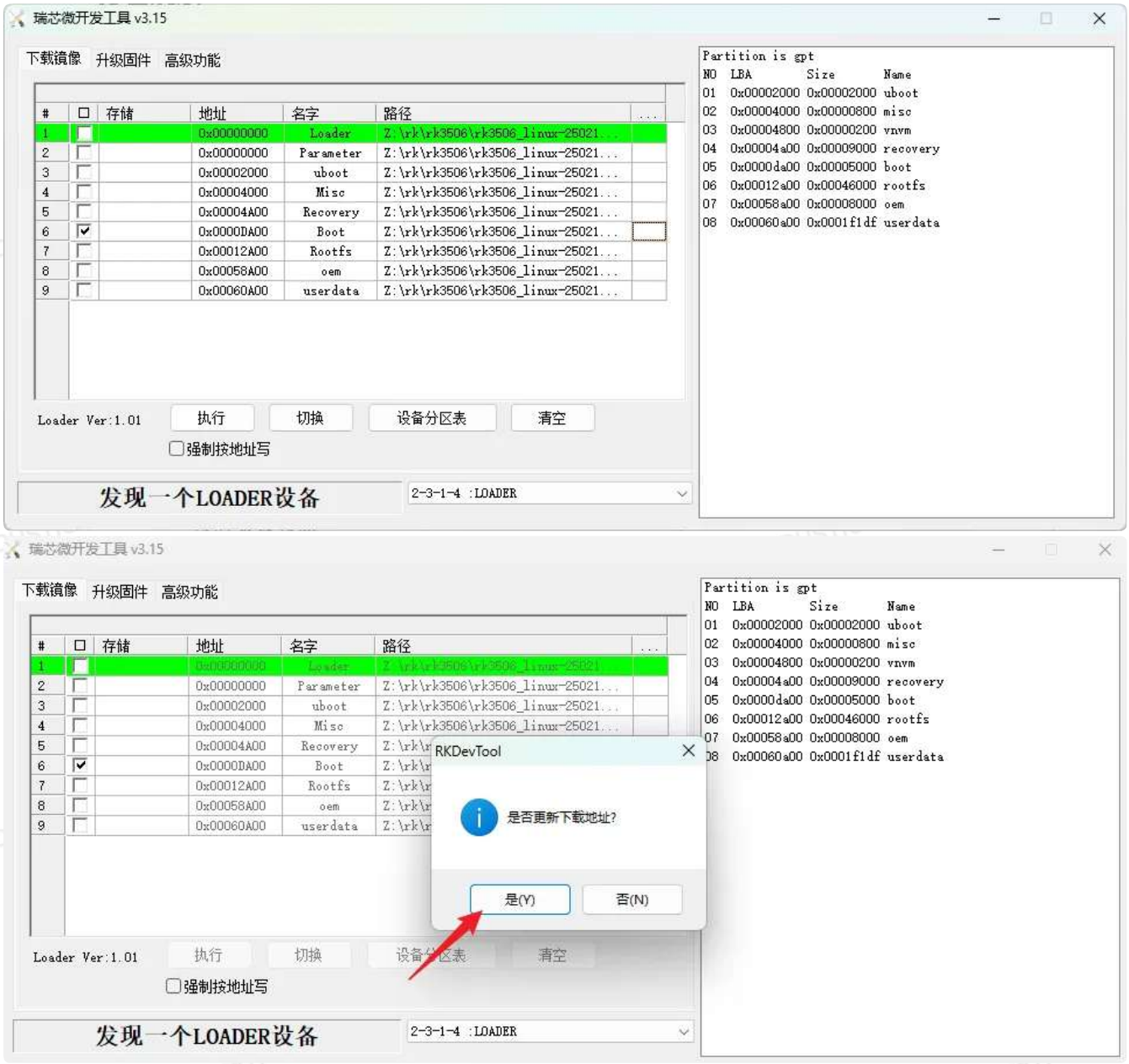
4.点击升级

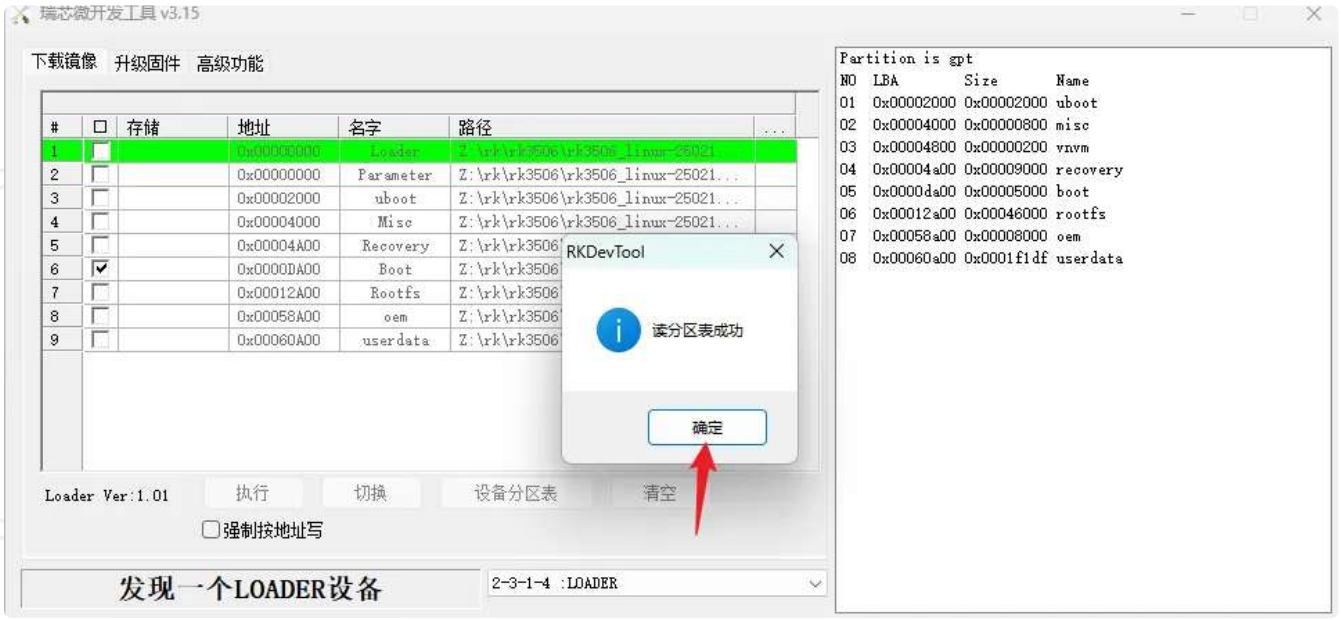


点击升级后，右侧会开始下载固件；下载完成后，板子会自动启动。

2.分包烧录

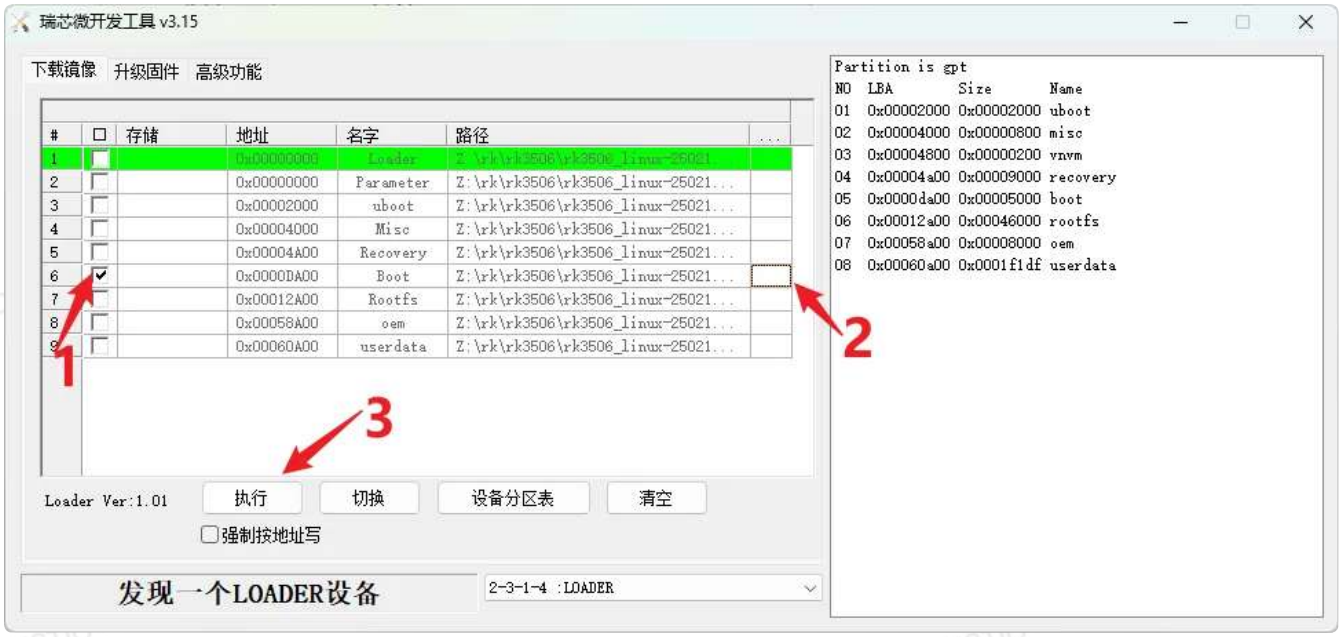
1.更新下载地址





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



7.驱动开发

1.CAN

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。

2.DTS 节点配置

公共配置：kernel/arch/arm/boot/dts/rk3506.dtsi

```Bash
can0: can@ff320000 {
      compatible = "rockchip,rk3506-canfd", "rockchip,rk3576-canfd";
      reg = <0xff320000 0x1000>;
      interrupts = <GIC_SPI 45 IRQ_TYPE_LEVEL_HIGH>;
      clocks = <&cru CLK_CAN0>, <&cru HCLK_CAN0>;
      clock-names = "baudclk", "apb_pclk";
      resets = <&cru SRST_CAN0>, <&cru SRST_H_CAN0>;
      reset-names = "can", "can-apb";
      assigned-clocks = <&cru CLK_CAN0>;
      assigned-clock-rates = <300000000>;
      status = "disabled";
    };

    can1: can@ff330000 {
      compatible = "rockchip,rk3506-canfd", "rockchip,rk3576-canfd";
      reg = <0xff330000 0x1000>;
      interrupts = <GIC_SPI 46 IRQ_TYPE_LEVEL_HIGH>;
      clocks = <&cru CLK_CAN1>, <&cru HCLK_CAN1>;
      clock-names = "baudclk", "apb_pclk";
      resets = <&cru SRST_CAN1>, <&cru SRST_H_CAN1>;
      reset-names = "can", "can-apb";
      assigned-clocks = <&cru CLK_CAN1>;
      assigned-clock-rates = <300000000>;
      status = "disabled";
    };
```

2.1 配置CAN1

板级配置：ido_evb3506_v1a-emmc.dtsi

```Bash
//和rgb冲突
&can0 {
  assigned-clocks = <&cru CLK_CAN0>;
  assigned-clock-rates = <200000000>;
  pinctrl-0 = <&rm_io25_can0_tx &rm_io26_can0_rx>;
  pinctrl-names = "default";
  status = "disabled";
};

&can1 {
  assigned-clocks = <&cru CLK_CAN1>;
  assigned-clock-rates = <200000000>;
  pinctrl-0 = <&rm_io17_can1_tx &rm_io18_can1_rx>;
  pinctrl-names = "default";
  status = "okay";
};
```

2.2 双CAN配置

板级配置：ido_evb3506_v1a-emmc.dtsi

```Bash
//和rgb冲突
&can0 {
  assigned-clocks = <&cru CLK_CAN0>;
  assigned-clock-rates = <200000000>;
  pinctrl-0 = <&rm_io25_can0_tx &rm_io26_can0_rx>;
  pinctrl-names = "default";
  status = "okay";
};

&can1 {
  assigned-clocks = <&cru CLK_CAN1>;
  assigned-clock-rates = <200000000>;
  pinctrl-0 = <&rm_io17_can1_tx &rm_io18_can1_rx>;
  pinctrl-names = "default";
  status = "okay";
};
```

板级配置：ido_evb3506_v1a-rgb-emmc.dts

```bash
&rgb {
    status = "disabled";
    pinctrl-names = "default";
    pinctrl-0 = <&rgb888_pins>;

    ports {
      rgb_out: port@1 {
        reg = <1>;
        #address-cells = <1>;
        #size-cells = <0>;

        rgb_out_panel: endpoint@0 {
          reg = <0>;
          remote-endpoint = <&panel_in_rgb>;
        };
      };
    };
  };

```

另外可设置时钟频率 `assigned-clock-rates`，如果 CAN 的比特率低于等于 3M 建议修改 CAN 时钟到 100M，信号更稳定。高于 3M 比特率的，时钟设置 200M 就可以。

3.通信测试

```bash
#检查can设备
$ ip link show

#在收发端关闭can0设备
$ ip link set can0 down

#设置仲裁段1M波特率，数据段1M波特率
$ ip link set can0 type can bitrate 1000000 dbitrate 1000000 fd on
[   63.834376] rk3576_canfd ff330000.can can0: bitrate error 0.3%
[   63.834512] rk3576_canfd ff330000.can can0: bitrate error 0.3%

#在收发端打开can0设备
$ ip link set can0 up

#在接收端执行candump,阻塞等待报文
$ candump can0

#在发送端执行cansend，发送报文
$ cansend can0 123#1122334455667788

#检查是否启用成功
$ ip link show can0
```


4. FAQs

总结调试过程中遇到的几个问题及解决方法：

1.报文发送后很久才接收到，或者接收不到。

检查总线 CAN_H 和 CAN_L， 杜邦线是否松动或者接反。

2.CAN时钟频率配置

如果CAN的比特率低于等于3M建议修改CAN时钟到100M,信号更稳定。高于3M比特率的，时钟设置200M就可以。

CAN时钟频率修改方法参考如下：

```bash
@@ -48,7 +48,7 @@ can0: can@ff320000 {
            resets = <&cru SRST_CAN0>, <&cru SRST_H_CAN0>;
            reset-names = "can", "can-apb";
            assigned-clocks = <&cru CLK_CAN0>;
-           assigned-clock-rates = <300000000>;
+           assigned-clock-rates = <200000000>;
            status = "disabled";
     };

@@ -61,7 +61,7 @@ can1: can@ff330000 {
            resets = <&cru SRST_CAN1>, <&cru SRST_H_CAN1>;
            reset-names = "can", "can-apb";
            assigned-clocks = <&cru CLK_CAN1>;
-           assigned-clock-rates = <300000000>;
+           assigned-clock-rates = <200000000>;
            status = "disabled";
     };
```

在某些时钟频率下，CAN的bitrate无法获得准确的速率，可以自行调整assigned-clock-rates去解决。

查看是否得到所需的bitrare：

```bash
ip -d link show can0
```

注意：evb3506的can0和RGB屏幕有冲突，默认dts中只打开can1节点，此节点在系统下注册成can0。

2.Ethernet

1.公共配置

`kernel/arch/arm/boot/dts/rk3506.dtsi`

```
gmac0: ethernet@ff4c8000 {
    compatible = "rockchip,rk3506-gmac", "snps,dwmac-4.20a";
    reg = <0xff4c8000 0x2000>;
    interrupts = <GIC_SPI 66 IRQ_TYPE_LEVEL_HIGH>,
           <GIC_SPI 69 IRQ_TYPE_LEVEL_HIGH>;
    interrupt-names = "macirq", "eth_wake_irq";
    rockchip,grf = <&grf>;
    clocks = <&cru CLK_MAC0>, <&cru CLK_MAC0_PTP>,
       <&cru PCLK_MAC0>, <&cru ACLK_MAC0>;
    clock-names = "stmmaceth", "ptp_ref",
           "pclk_mac", "aclk_mac";
    resets = <&cru SRST_A_MAC0>;
    reset-names = "stmmaceth";

    snps,mixed-burst;
    snps,tso;

    snps,axi-config = <&gmac0_stmmac_axi_setup>;
    snps,mtl-rx-config = <&gmac0_mtl_rx_setup>;
    snps,mtl-tx-config = <&gmac0_mtl_tx_setup>;

    phy-mode = "rmii";
    status = "disabled";

    mdio0: mdio {
      compatible = "snps,dwmac-mdio";
      #address-cells = <0x1>;
      #size-cells = <0x0>;
    };

    gmac0_stmmac_axi_setup: stmmac-axi-config {
      snps,wr_osr_lmt = <4>;
      snps,rd_osr_lmt = <8>;
      snps,blen = <0 0 0 0 16 8 4>;
    };

    gmac0_mtl_rx_setup: rx-queues-config {
      snps,rx-queues-to-use = <1>;
      queue0 {
        status = "okay";
      };
    };

    gmac0_mtl_tx_setup: tx-queues-config {
      snps,tx-queues-to-use = <1>;
```

```
      queue0 {
        status = "okay";
      };
    };
  };

  gmac1: ethernet@ff4d0000 {
    compatible = "rockchip,rk3506-gmac", "snps,dwmac-4.20a";
    reg = <0xff4d0000 0x2000>;
    interrupts = <GIC_SPI 70 IRQ_TYPE_LEVEL_HIGH>,
           <GIC_SPI 73 IRQ_TYPE_LEVEL_HIGH>;
    interrupt-names = "macirq", "eth_wake_irq";
    rockchip,grf = <&grf>;
    clocks = <&cru CLK_MAC1>, <&cru CLK_MAC1_PTP>,
       <&cru PCLK_MAC1>, <&cru ACLK_MAC1>;
    clock-names = "stmmaceth", "ptp_ref",
           "pclk_mac", "aclk_mac";
    resets = <&cru SRST_A_MAC1>;
    reset-names = "stmmaceth";

    snps,mixed-burst;
    snps,tso;

    snps,axi-config = <&gmac1_stmmac_axi_setup>;
    snps,mtl-rx-config = <&gmac1_mtl_rx_setup>;
    snps,mtl-tx-config = <&gmac1_mtl_tx_setup>;

    phy-mode = "rmii";
    status = "disabled";

    mdio1: mdio {
      compatible = "snps,dwmac-mdio";
      #address-cells = <0x1>;
      #size-cells = <0x0>;
    };

    gmac1_stmmac_axi_setup: stmmac-axi-config {
      snps,wr_osr_lmt = <4>;
      snps,rd_osr_lmt = <8>;
      snps,blen = <0 0 0 0 16 8 4>;
    };

    gmac1_mtl_rx_setup: rx-queues-config {
      snps,rx-queues-to-use = <1>;
      queue0 {
        status = "okay";
      };
    };
```

```
94
95        gmac1_mtl_tx_setup: tx-queues-config {
96          snps,tx-queues-to-use = <1>;
97          queue0 {
98            status = "okay";
99          };
100       };
101     };
```

2.板级的配置

```Bash
&gmac0 {
  phy-mode = "rmii";
  clock_in_out = "input";

  snps,reset-gpio = <&gpio0 RK_PC3 GPIO_ACTIVE_LOW>;
  snps,reset-active-low;
  snps,reset-delays-us = <0 20000 100000>;

  pinctrl-names = "default";
  pinctrl-0 = <&eth_rmii0_miim_pins
         &eth_rmii0_tx_bus2_pins
         &eth_rmii0_rx_bus2_pins
         &eth_rmii0_clk_pins>;

  phy-handle = <&rmii_phy0>;
  status = "okay";
};

&gmac1 {
  phy-mode = "rmii";
  clock_in_out = "input";

  snps,reset-gpio = <&gpio0 RK_PC4 GPIO_ACTIVE_LOW>;
  snps,reset-active-low;
  snps,reset-delays-us = <0 20000 100000>;

  pinctrl-names = "default";
  pinctrl-0 = <&eth_rmii1_miim_pins
         &eth_rmii1_tx_bus2_pins
         &eth_rmii1_rx_bus2_pins
         &eth_rmii1_clk_pins>;

  phy-handle = <&rmii_phy1>;
  status = "okay";
};

```

3.LCD

IDO-EVB3506-V1 有一路 MIPI DSI 显示输出接口，支持 2 Lane 的数据输出，最高可输出 720x720@60Hz。

1.dts修改

Bash

```
// SPDX-License-Identifier: (GPL-2.0+ OR MIT)
/*
 * Copyright (c) 2024 Rockchip Electronics Co., Ltd.
 */


#include <dt-bindings/display/drm_mipi_dsi.h>
#include <dt-bindings/input/rk-input.h>
#include <dt-bindings/suspend/rockchip-rk3506.h>
#include "rk3502-ido_evb3506_v1a.dtsi"

/ {
  backlight: backlight {
    compatible = "pwm-backlight";
    pwms = <&pwm0_4ch_2 0 25000 0>;
    brightness-levels = <
        0  20  20  21  21  22  22  23
       23  24  24  25  25  26  26  27
       27  28  28  29  29  30  30  31
       31  32  32  33  33  34  34  35
       35  36  36  37  37  38  38  39
       40  41  42  43  44  45  46  47
       48  49  50  51  52  53  54  55
       56  57  58  59  60  61  62  63
       64  65  66  67  68  69  70  71
       72  73  74  75  76  77  78  79
       80  81  82  83  84  85  86  87
       88  89  90  91  92  93  94  95
       96  97  98  99 100 101 102 103
      104 105 106 107 108 109 110 111
      112 113 114 115 116 117 118 119
      120 121 122 123 124 125 126 127
      128 129 130 131 132 133 134 135
      136 137 138 139 140 141 142 143
      144 145 146 147 148 149 150 151
      152 153 154 155 156 157 158 159
      160 161 162 163 164 165 166 167
      168 169 170 171 172 173 174 175
      176 177 178 179 180 181 182 183
      184 185 186 187 188 189 190 191
      192 193 194 195 196 197 198 199
      200 201 202 203 204 205 206 207
      208 209 210 211 212 213 214 215
      216 217 218 219 220 221 222 223
      224 225 226 227 228 229 230 231
```

```
        232 233 234 235 236 237 238 239
        240 241 242 243 244 245 246 247
        248 249 250 251 252 253 254 255
      >;
      default-brightness-level = <200>;
      status = "okay";
    };
};


&display_subsystem {
  logo-memory-region = <&drm_logo>;
  status = "okay";
};

&pwm0_4ch_2 {
        pinctrl-names = "active";
        pinctrl-0 = <&rm_io4_pwm0_ch2>;
        status = "okay";
};

&dsi {
  status = "okay";
  //rockchip,lane-rate = <850>;
  dsi_panel: panel@0 {
    status = "okay";
    compatible = "simple-panel-dsi";
    reg = <0>;
    backlight = <&backlight>;
    reset-gpios  = <&gpio0 RK_PB5 GPIO_ACTIVE_LOW>;
    prepare-delay-ms = <5>;
    reset-delay-ms = <1>;
    init-delay-ms = <80>;
    disable-delay-ms = <10>;
    unprepare-delay-ms = <5>;

    width-mm = <68>;
    height-mm = <121>;

    dsi,flags = <(MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
            MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_NO_EOT_PACKET)>;
    dsi,format = <MIPI_DSI_FMT_RGB888>;
    dsi,lanes  = <2>;
    panel-init-sequence = [
      39 00 04 B9 F1 12 87
      39 00 04 B2 B4 03 70
      39 00 0B B3 10 10 28 28 03 FF 00 00 00 00
      15 00 02 B4 80
```

```
        39 00 03 B5 0A 0A
        39 00 03 B6 8D 8D
        39 00 05 B8 26 22 F0 13
        39 00 1C BA 31 81 05 F9 0E 0E 20 00 00 00 00 00 00 00 44 25 00 91 0
A 00 00 01 4F 01 00 00 37
        15 00 02 BC 47
        39 00 06 BF 02 10 00 80 04
        39 00 0A C0 73 73 50 50 00 00 12 73 00
        39 00 12 C1 36 00 32 32 77 E1 77 77 CC CC FF FF 11 11 00 00 32
        39 00 0D C7 10 00 0A 00 00 00 00 00 ED C5 00 A5
        39 00 05 C8 10 40 1E 03
        15 00 02 CC 0B
        39 00 23 E0 00 0A 0F 2A 33 3F 44 39 06 0C 0E 14 15 13 15 10 18 00 0
A 0F 2A 33 3F 44 39 06 0C 0E 14 15 13 15 10 18
        39 00 08 E1 11 11 91 00 00 00 00
        39 00 0F E3 07 07 0B 0B 0B 0B 00 00 00 00 FF 04 C0 10
        39 00 40 E9 C8 10 0A 00 00 80 81 12 31 23 4F 86 A0 00 47 08 00 00 0
C 00 00 00 00 00 0C 00 00 00 98 02 8B AF 46 02 88 88 88 88 88 98 13 8B A
F 57 13 88 88 88 88 88 00 00 00 00 00 00 00 00 00 00 00 00 00 00
        39 00 3E EA 97 0C 09 09 09 78 00 00 00 00 00 00 9F 31 8B A8 31 75 8
8 88 88 88 88 9F 20 8B A8 20 64 88 88 88 88 88 23 00 00 02 71 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 40 80 81 00 00 00 00
        39 00 04 EF FF FF 01
        05 96 01 11
        05 14 01 29
      ];

      disp_timings0: display-timings {
        native-mode = <&dsi_timing0>;
        dsi_timing0: timing0 {
          clock-frequency = <48000000>;
          hactive = <720>;
          vactive = <720>;
          hfront-porch = <106>;
          hsync-len = <8>;
          hback-porch = <120>;
          vfront-porch = <20>;
          vsync-len = <4>;
          vback-porch = <20>;
          hsync-active = <0>;
          vsync-active = <0>;
          de-active = <0>;
          pixelclk-active = <0>;
        };
      };

      ports {
        #address-cells = <1>;
```

```
        #size-cells = <0>;

        port@0 {
          reg = <0>;
          panel_in_dsi: endpoint {
            remote-endpoint = <&dsi_out_panel>;
          };
        };
      };
    };

    ports {
      #address-cells = <1>;
      #size-cells = <0>;

      port@1 {
        reg = <1>;
        dsi_out_panel: endpoint {
          remote-endpoint = <&panel_in_dsi>;
        };
      };
    };

};

&dsi_dphy {
    status = "okay";
};

&dsi_in_vop {
  status = "okay";
};

&dsi_panel {
  power-supply = <&vcc3v3_lcd_n>;
};

&route_dsi{
        status = "okay";
    connect = <&vop_out_dsi>;
};

&i2c0 {
  status = "okay";
  pinctrl-names = "default";
  pinctrl-0 = <&rm_io0_i2c0_scl
         &rm_io1_i2c0_sda>;
```

```
    gt911@14 {
      compatible = "goodix,gt9xx";
      reg = <0x14>;
      pinctrl-0 = <&touch_gpio>;
      touch-gpio = <&gpio0 RK_PA2 IRQ_TYPE_LEVEL_LOW>;
      reset-gpio = <&gpio0 RK_PA3 GPIO_ACTIVE_HIGH>;
      max-x = <720>;
      max-y = <720>;
      tp-size = <9110>;
      status = "okay";
    };
  };

  &vop {
    status = "okay";
  };

  &pinctrl {
    touch {
      touch_gpio: touch-gpio {
        rockchip,pins = <0 RK_PA2 RK_FUNC_GPIO &pcfg_pull_none>;
      };

      reset_gpio: reset-gpio {
        rockchip,pins = <0 RK_PA3 RK_FUNC_GPIO &pcfg_pull_none>;
      };
    };
  };
```

2.注意事项

在配置 MIPI DSI的时候，如果出现异常现象，如黑屏，画面拉伸，显示有噪点等，需要注意排查：

1. 显示时序是否配置正确，尤其是 DCLK 的配置。
2. 上下电时序是否正确，在文件 `kernel/drivers/gpu/drm/panel/panel-simple.c` 中的 `panel_simple_prepare` 和 `panel_simple_unprepare` 函数内，调用了设备树中所配置的上下电时序和 gpio 口。

如果选择在 uboot 阶段开启开机 logo, 那么还需要排查 `u-boot/drivers/video/drm/rockchip_panel.c` 文件内的 `panel_simple_prepare` 和 `panel_simple_unprepare` 函数。

3. 搭示波器看一下上电时序是否正确，主要是确认 LCD 使能引脚、复位引脚和屏幕上电指令间的时序是否正确。

3.调试方法

获取系统中正在使用的 Video Port（与所连接的显示控制器） 信息：

```Bash
root@rk3506-buildroot:/# cat /sys/kernel/debug/dri/0/summary
VOP [ff600000.vop]: ACTIVE
    Connector: DSI-1
  bus_format[100a]: RGB888_1X24
  overlay_mode[0] output_mode[0]color-encoding[1] color-range[1]
    Display mode: 720x720p66
  dclk[48000 kHz] real_dclk[47536 kHz] aclk[294912 kHz] type[48] flag[a]
  H: 720 826 834 954
  V: 720 740 744 764
    win1-0: ACTIVE
  format: BG24 little-endian (0x34324742) SDR[0] color-encoding[0] color-range[0]
  csc: y2r[0] r2r[0] r2y[0] csc mode[0]
  zpos: 0
  src: pos[0x0] rect[720x720]
  dst: pos[0x0] rect[720x720]
  buf[0]: addr: 0x0f000000 pitch: 2304 offset: 0
    post: sdr2hdr[0] hdr2sdr[0]
    pre : sdr2hdr[0]
    post CSC: r2y[0] y2r[0] CSC mode[…]
```

一般如果遇到屏幕无法显示的问题，都需要先执行上面的命令去看一下连接状态和分辨率是否正确。

注意：rk3506linux sdk只有一个显示接口、可以支持RGB/MIPI两种接口、但是同时只能使用一个。

4.RTC

IDO-EVB3506-V1 采用 HYM8563 作为RTC(*Real Time Clock*)，需要接入 RTC 电池给 RTC 芯片供电才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/arm/boot/dts/rk3502-ido_evb3506_v1a.dtsi`

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

Linux下的rtc使用方法为：

```bash
#同步网络时间
$ ntpdate cn.pool.ntp.org

#查看当前 RTC 的日期和时间:
$ cat /sys/class/rtc/rtc0/date
2021-01-01

$ cat /sys/class/rtc/rtc0/time
17:18:14

$ 查看系统时间
$ date
Wed Mar 12 18:46:59 CST 2025

#设置系统时间
$ date -s "2024-10-09 14:02:30"

#将rtc时间调整为与目前的系统时间一致
$ hwclock -w

#获取硬件rtc当前时间（断电重启读取时间没有太大偏差）
$ hwclock -r
2024-10-09 14:02:35.945604+00:00

#将rtc时间设置为系统时间
hwclock --systohc
```

5.UART

1.DTS配置

```bash
&uart1 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&rm_io15_uart1_tx &rm_io16_uart1_rx>;
};
```

配置好串口后，硬件接口对应软件上的节点为：

```
UART1:   /dev/ttyS1
```

2.收发测试

使用microcom可以进行串口收发测试，命令如下：

```
root@rk3506-buildroot:/# microcom -s 115200 /dev/ttyS1
```

注意：测试完成，按Ctrl+x退出。

8.其它使用手册

1.Qt5使用

详情参考《IDO-EVB3506-V1 QT应用开发手册》.pdf

2.MQTT使用

详情参考《IDO-EVB3506-V1 MQTT应用案例》.pdf

3.GDB使用

详情参考《IDO-EVB3506-V1 GDB开发手册》.pdf

4.Docker使用

详情参考《IDO-EVB3506-V1 DOCKER容器安装使用手册》.pdf

5. LVGL使用

详情参考《IDO-EVB3506-V1 LVGL应用开发手册》.pdf

6.RT-Linux使用

详情参考《IDO-EVB3506-V1 RT-Linux使用手册》.pdf