

IDO-EVB3588S-V1 Linux开发手册



IDO-EVB3588S-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	修订内容	修订	审核	日期
----	------	----	----	----

V1.0	创建文档	MHK	IDO	2025/08/08
------	------	-----	-----	------------

1. 上手指南

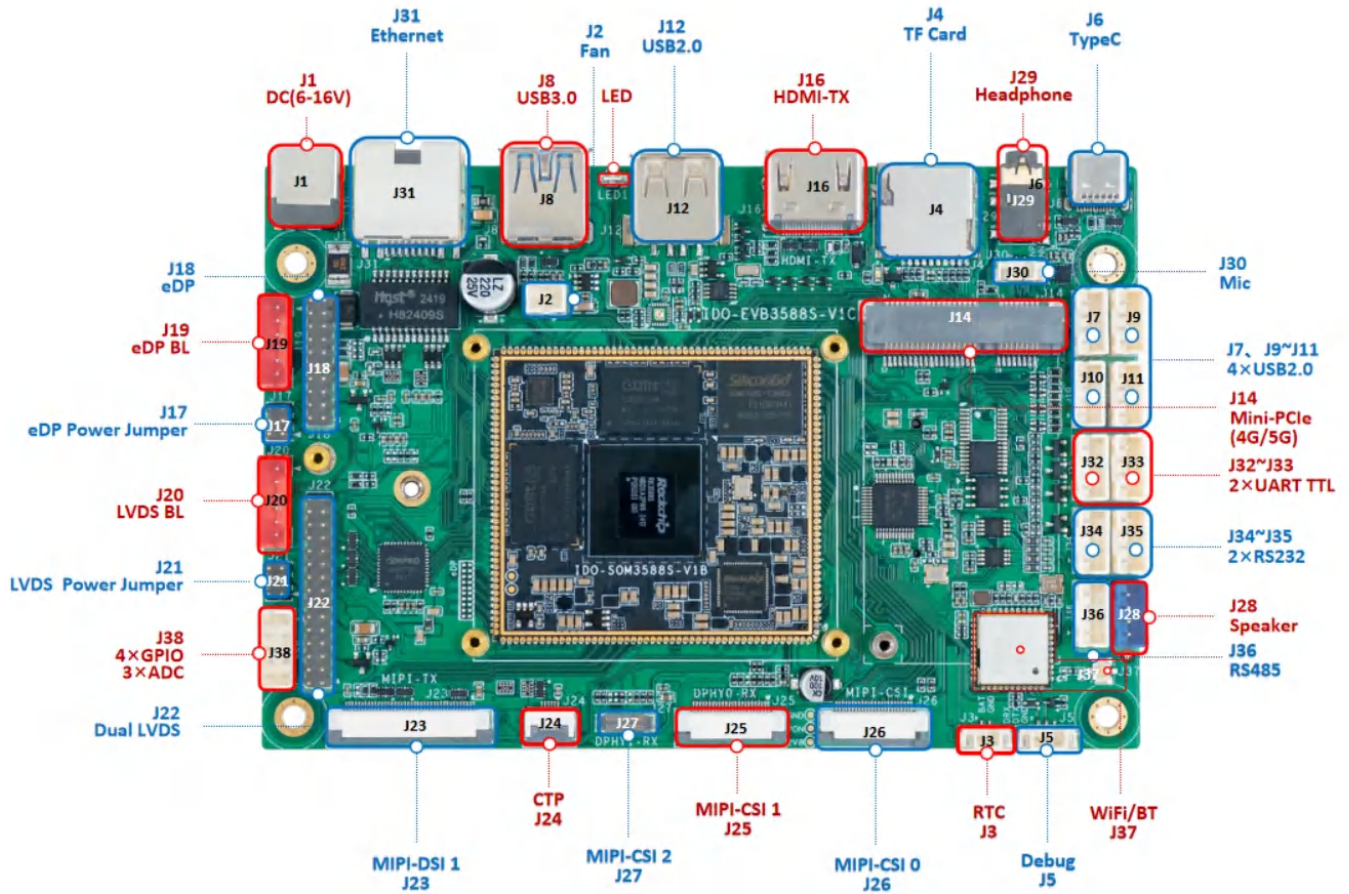
1.1. 介绍

IDO-EVB3588S-V1具有丰富的视频输入输出接口（HDMI-TX/eDP1.3/MIPI-DSI/MIPI-CSI/LVDS），高速通信接口（Enthernet/PCIe2.0/USB3.0/TF卡），工业互联接口

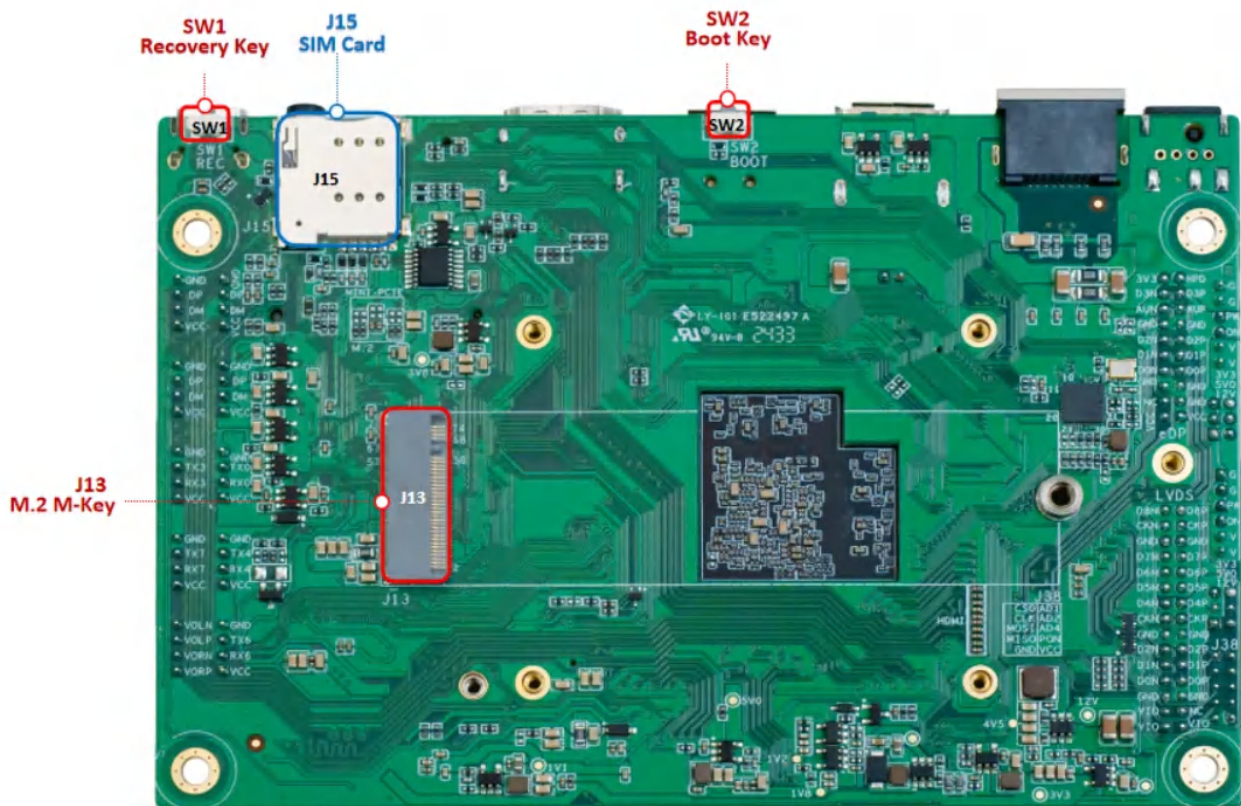
（RS485/RS232/UART），可作为RK3588S开发评估板，也普遍适用于各种智慧显示终端产品、视频类终端产品、工业自动化终端产品和边缘计算网关类产品。应用可覆盖边缘计算、人工智能、工业HMI、工业网关、智慧医疗、自助终端、智能零售、能源电力等行业

IDO-EVB3588S 配置 Rockchip 八核 64 位 AIOT 处理器，采用了 8nm LP 制程；搭载八核（Cortex-A76 x 4 + Cortex-A55 x 4）64位 CPU，主频高达2.4 GHz。

集成 ARM Mali-G610 MP4 四核 GPU，内置 AI 加速器 NPU，可提供6 Tops 算力，支持主流的深度学习框架；最大支持32 GB 大内存；支持 8K 视频编解码和多种格式的视频输入输出；支持多种操作系统；可适用于 ARM PC、边缘计算、云服务器、智能NVR 等领域。



IDO-EVB3588S-V1C 正面



IDO-EVB3588-V1C 背面

2. Linux开发

2.1. 获取 SDK

下载路径：百度网盘/Linux/软件资料/SDK/

2.2. 校验md5值

```
1 $ md5sum rk3588_linux6.1_rkr6_v1.tar.gz
```

2.3. 解压

```
1  $ tar -xvf rk3588_linux6.1_rkr3_v1.tar.gz
2
3  $ cd rk3588_linux6.1_rkr3_v1
4
5  #获取代码
6  $ git reset --hard
7
8  #查看分支(默认分支为ido-evb3588)
9  $ git branch
10 * ido-evb3588
11     ido-evb3588-preempt-rt
12     ido-evb3588-ubuntu
13     ido-evb3588-ubuntu-preempt-rt
14     ido-evb3588-xenomai-rt
15     ido-evb3588s
16     ido-evb3588s-preempt-rt
17     ido-evb3588s-ubuntu
18     ido-evb3588s-ubuntu-preempt-rt
19     ido-evb3588s-xenomai-rt
20
21 #切换到EVB3588S分支
22 $ git checkout ido-evb3588s
```

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/rk3588` 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_ido-evb3588s_v1*_defconfig 作为基础的配置。其他版型的配置文件会引用 rockchip_rk3588_ido-evb3588_v1b*_defconfig 并且覆盖其中的一些变量配置。以 rockchip_ido-evb3588s_v1_hdmi_defconfig 做介绍：

```
1  RK_BUILDROOT_BASE_CFG="rk3588"
2  RK_CHIP_FAMILY="rk3588"                #芯片类型
3  # RK_WIFIBT is not set
4  RK_UBOOT_SPL=y
5  RK_KERNEL_DTS_NAME="ido-evb3588s-v1b-hdmi" # 板型 dts 名称
6  RK_USE_FIT_IMG=y
7  RK_KERNEL_CFG="rockchip_linux_defconfig" # kernel 编译配置
```

2.6. 分区说明

parameter.txt 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: RK3588
3  MACHINE_ID: 007
4  MANUFACTURER: RK3588
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),0x00040000@0x00078000(oem),0x00400000@0x000B8000(userdata),-@0x004B8000(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


```
▼ Bash |
1 $ ./build.sh lunch
2 Log colors: message notice warning error fatal
3
4 Log saved at /mnt/5c953a98-9ee6-45b9-8cea-a2898eb313d7/mhk/rk/rk3588/linux/rk3588_linux5.10/rk3588_linux_rkr9_250722/output/sessions/2025-08-16_17-57-25
5 Pick a defconfig:
6
7 1. rockchip_ido_evb3588s_v1_edp_1920x1080_defconfig
8 2. rockchip_ido_evb3588s_v1_hdmi_defconfig
9 3. rockchip_ido_evb3588s_v1_lvds_1920x1080_defconfig
10 4. rockchip_ido_evb3588s_v1_mipi_1920x1200_defconfig
11 ▾ Which would you like? [1]:
```

选择对应的数字即可。

3.2. 全部编译

```
▼ Bash |
1 $ ./build.sh
```

固件编译后生成路径：rockdev/update.img

3.3. 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
```

注意：编译ubuntu需要切换ubuntu分支。

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

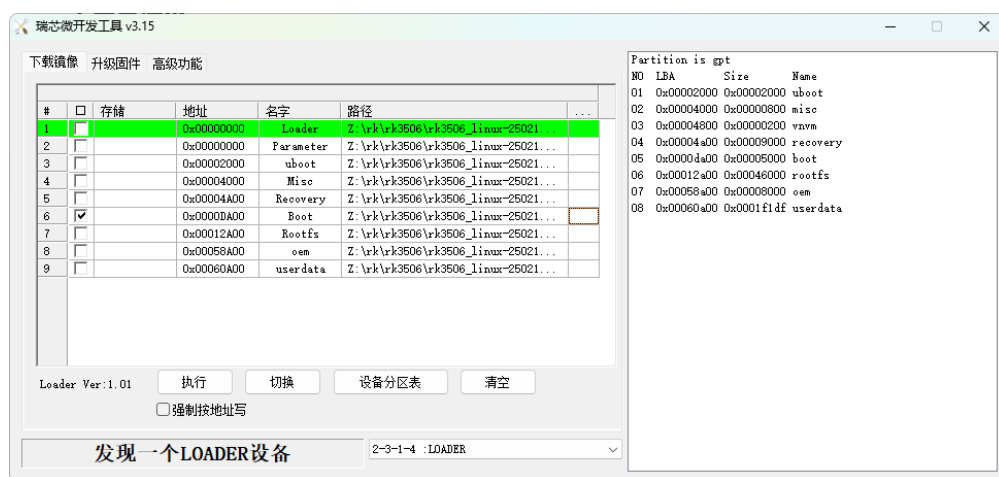
3.3.7. 固件打包

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

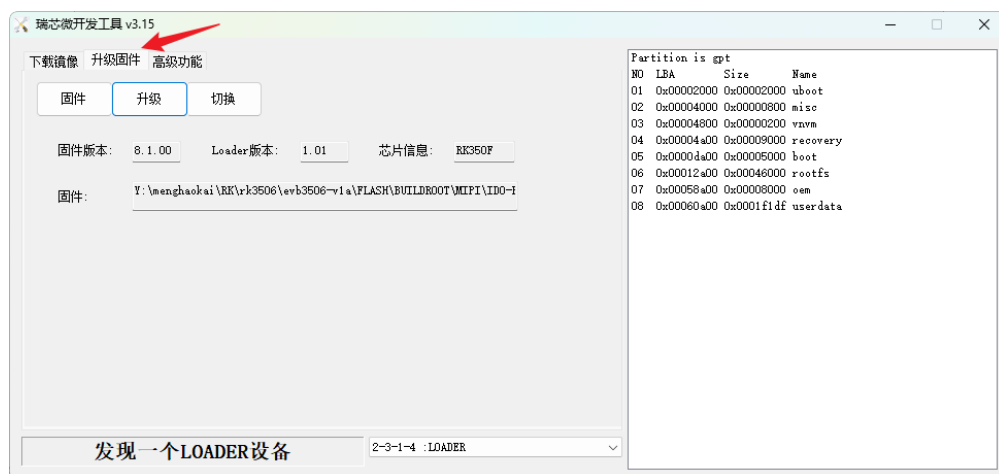
4. 烧录说明

4.1. 整包烧录

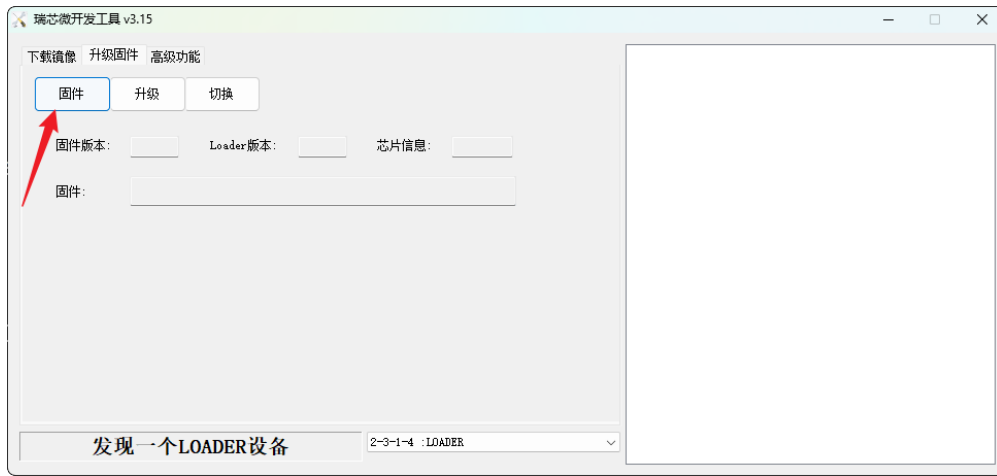
4.1.1. 运行RK开发工具



4.1.2. 点击升级固件

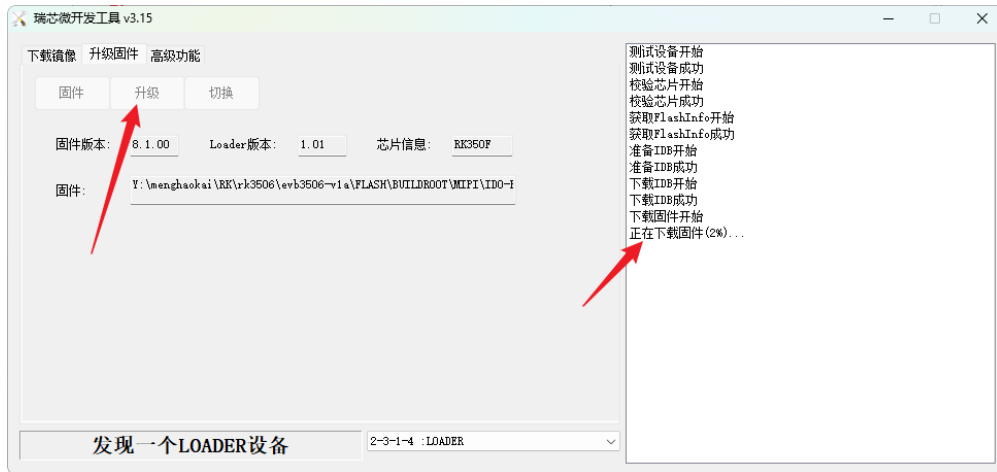


4.1.3. 点击固件



选择：rockdev/update.img

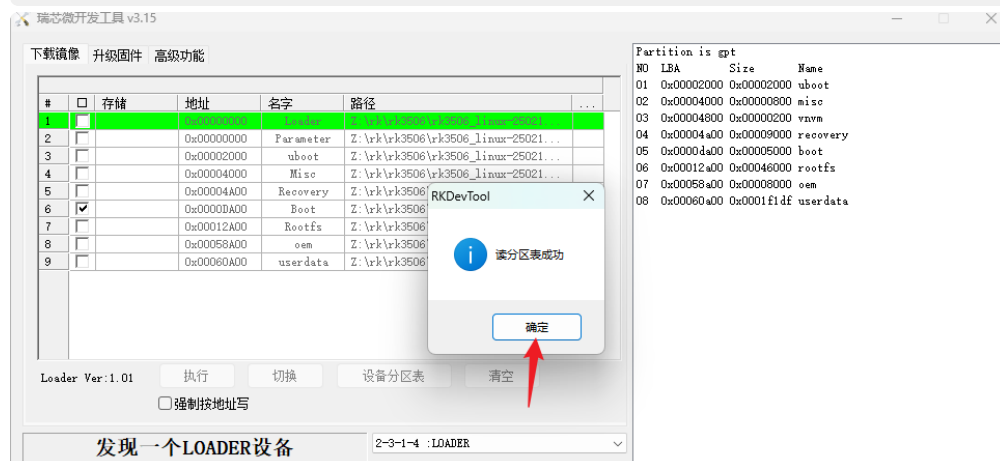
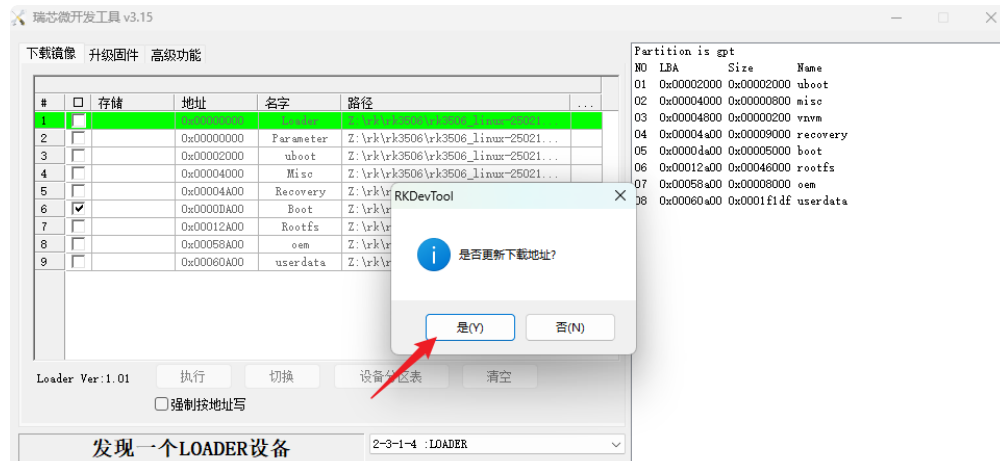
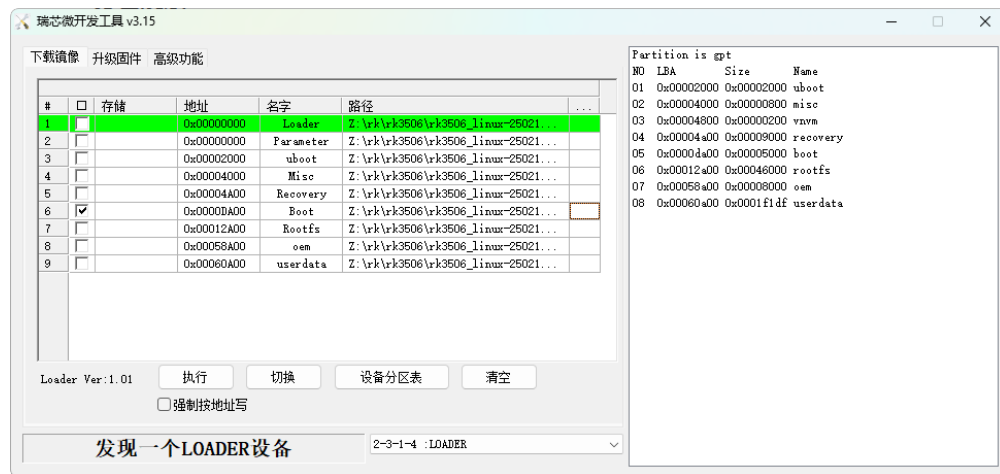
4.1.4. 点击升级



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。

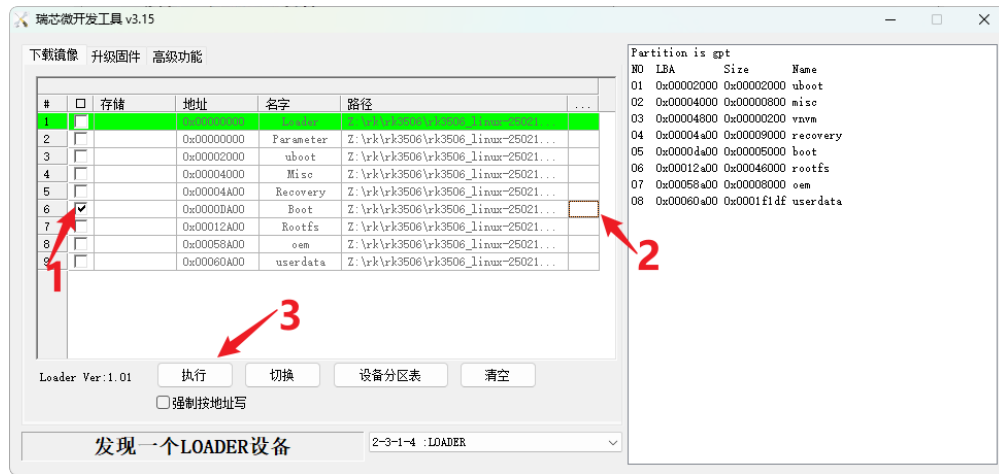
4.2. 分包烧录

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. Ethernet

5.1.1.1. 公共配置

kernel/arch/arm64/boot/dts/rk3588s.dtsi

```

1  ▼ gmac1: ethernet@fe1c0000 {
2      compatible = "rockchip,rk3588-gmac", "snps,dwmac-4.20a";
3      reg = <0x0 0xfe1c0000 0x0 0x10000>;
4      interrupts = <GIC_SPI 234 IRQ_TYPE_LEVEL_HIGH>,
5                  <GIC_SPI 233 IRQ_TYPE_LEVEL_HIGH>;
6      interrupt-names = "macirq", "eth_wake_irq";
7      rockchip,grf = <&sys_grf>;
8      rockchip,php_grf = <&php_grf>;
9      clocks = <&cru CLK_GMAC_125M>, <&cru CLK_GMAC_50M>,
10             <&cru PCLK_GMAC1>, <&cru ACLK_GMAC1>,
11             <&cru CLK_GMAC1_PTP_REF>;
12      clock-names = "stmmaceth", "clk_mac_ref",
13                   "pclk_mac", "aclk_mac",
14                   "ptp_ref";
15      resets = <&cru SRST_A_GMAC1>;
16      reset-names = "stmmaceth";
17      power-domains = <&power RK3588_PD_GMAC>;
18
19      snps,mixed-burst;
20      snps,tso;
21
22      snps,axi-config = <&gmac1_stmmac_axi_setup>;
23      snps,mtl-rx-config = <&gmac1_mtl_rx_setup>;
24      snps,mtl-tx-config = <&gmac1_mtl_tx_setup>;
25      status = "disabled";
26
27  ▼ mdio1: mdio {
28      compatible = "snps,dwmac-mdio";
29      #address-cells = <0x1>;
30      #size-cells = <0x0>;
31  };
32
33  ▼ gmac1_stmmac_axi_setup: stmmac-axi-config {
34      snps,wr_osr_lmt = <4>;
35      snps,rd_osr_lmt = <8>;
36      snps,blen = <0 0 0 0 16 8 4>;
37  };
38
39  ▼ gmac1_mtl_rx_setup: rx-queues-config {
40      snps,rx-queues-to-use = <1>;
41      queue0 {};
42  };
43
44  ▼ gmac1_mtl_tx_setup: tx-queues-config {
45      snps,tx-queues-to-use = <1>;

```

```

46     queue0 {};
47 };
48 };
49 };

```

板级的配置

```

1 ▾ &gmac1 {
2     /* Use rgmii-rxid mode to disable rx delay inside Soc */
3     //phy-mode = "rgmii-rxid";
4     phy-mode = "rgmii";
5     clock_in_out = "input";
6
7     snps,reset-gpio = <&gpio1 RK_PB0 GPIO_ACTIVE_LOW>;
8     snps,reset-active-low;
9     /* Reset time is 20ms, 100ms for rtl8211f */
10    snps,reset-delays-us = <0 20000 100000>;
11
12    pinctrl-names = "default";
13    pinctrl-0 = <&gmac1_miim
14        &gmac1_tx_bus2
15        &gmac1_rx_bus2
16        &gmac1_rgmii_clk
17        &gmac1_rgmii_bus
18        &gmac1_clkinout>;
19
20    tx_delay = <0x34>;
21    rx_delay = <0x32>;
22
23    phy-handle = <&rgmii_phy>;
24    status = "okay";
25 };
26
27 ▾ &mdio1 {
28 ▾ rgmii_phy: phy@1 {
29     compatible = "ethernet-phy-ieee802.3-c22";
30     reg = <0x01>;
31 };
32 };
33

```

5.3.LCD

打开 LVDS

```

1 ▾ / {
2 ▾   backlight: backlight {
3     compatible = "pwm-backlight";
4     brightness-levels = <
5         0  20  20  21  21  22  22  23
6         23  24  24  25  25  26  26  27
7         27  28  28  29  29  30  30  31
8         31  32  32  33  33  34  34  35
9         35  36  36  37  37  38  38  39
10        40  41  42  43  44  45  46  47
11        48  49  50  51  52  53  54  55
12        56  57  58  59  60  61  62  63
13        64  65  66  67  68  69  70  71
14        72  73  74  75  76  77  78  79
15        80  81  82  83  84  85  86  87
16        88  89  90  91  92  93  94  95
17        96  97  98  99 100 101 102 103
18       104 105 106 107 108 109 110 111
19       112 113 114 115 116 117 118 119
20       120 121 122 123 124 125 126 127
21       128 129 130 131 132 133 134 135
22       136 137 138 139 140 141 142 143
23       144 145 146 147 148 149 150 151
24       152 153 154 155 156 157 158 159
25       160 161 162 163 164 165 166 167
26       168 169 170 171 172 173 174 175
27       176 177 178 179 180 181 182 183
28       184 185 186 187 188 189 190 191
29       192 193 194 195 196 197 198 199
30       200 201 202 203 204 205 206 207
31       208 209 210 211 212 213 214 215
32       216 217 218 219 220 221 222 223
33       224 225 226 227 228 229 230 231
34       232 233 234 235 236 237 238 239
35       240 241 242 243 244 245 246 247
36       248 249 250 251 252 253 254 255
37     >;
38     default-brightness-level = <200>;
39   };
40 };
41
42 ▾ &backlight {
43     pwms = <&pwm12 0 25000 0>;
44     enable-gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_HIGH>;
45     status = "okay";

```

```

46 };
47
48 &pwm12 {
49     status = "okay";
50     pinctrl-0 = <&pwm12m1_pins>;
51 };
52
53 &dsi0 {
54     status = "okay";
55     //rockchip, lane-rate = <1000>;
56     dsi0_panel: panel@0 {
57         status = "okay";
58         compatible = "simple-panel-dsi";
59         reg = <0>;
60         backlight = <&backlight>;
61         reset-delay-ms = <10>;
62         enable-delay-ms = <10>;
63         prepare-delay-ms = <10>;
64         unprepare-delay-ms = <10>;
65         disable-delay-ms = <60>;
66         dsi, flags = <(MIPI_DSI_MODE_VIDEO | MIPI_DSI_MODE_VIDEO_BURST |
67             MIPI_DSI_MODE_LPM | MIPI_DSI_MODE_EOT_PACKET)>;
68         dsi, format = <MIPI_DSI_FMT_RGB888>;
69         dsi, lanes = <4>;
70         panel-init-sequence = [
71             29 00 02 27 AA
72             29 00 02 48 02
73             29 00 02 B6 20
74             29 00 02 01 80
75             29 00 02 02 38
76             29 00 02 03 47
77             29 00 02 04 84//hfp
78             29 00 02 05 32//hsync
79             29 00 02 06 58//hbp
80             29 00 02 07 00
81             29 00 02 08 04//vfp
82             29 00 02 09 20//vsync
83             29 00 02 0A 09//vbp
84             29 00 02 0B 82
85             29 00 02 0C 12
86             29 00 02 0D 01
87             29 00 02 0E 80
88             29 00 02 0F 20
89             29 00 02 10 20
90             29 00 02 11 03
91             29 00 02 12 1B
92             29 00 02 13 63//53 //寄偶
93             29 00 02 14 01

```

```

94         29 00 02 15 23
95         29 00 02 16 40
96         29 00 02 17 00
97         29 00 02 18 01
98         29 00 02 19 23
99         29 00 02 1A 40
100        29 00 02 1B 00
101        29 00 02 1E 46
102        29 00 02 51 30
103        29 00 02 1F 10
104        //29 00 02 2A 4D //彩条
105        29 00 02 2A 01
106        05 78 01 11
107        05 1E 01 29
108    ];
109
110    panel-exit-sequence = [
111        05 00 01 28
112        05 00 01 10
113    ];
114
115    disp_timings0: display-timings {
116        native-mode = <&dsi0_timing0>;
117        dsi0_timing0: timing0 {
118            clock-frequency = <148500000>;
119            hactive = <1920>;
120            vactive = <1080>;
121
122            hfront-porch = <0x84>;
123            hsync-len = <0x3c>;
124            hback-porch = <0x58>;
125
126            vfront-porch = <0x4>;
127            vsync-len = <0x20>;
128            vback-porch = <0x09>;
129
130            hsync-active = <0>;
131            vsync-active = <0>;
132            de-active = <0>;
133            pixelclk-active = <0>;
134        };
135    };
136
137    ports {
138        #address-cells = <1>;
139        #size-cells = <0>;
140
141        port@0 {

```

```

142         reg = <0>;
143     panel_in_dsi: endpoint {
144         remote-endpoint = <&dsi_out_panel>;
145     };
146 };
147 };
148 };
149 };
150 ports {
151     #address-cells = <1>;
152     #size-cells = <0>;
153
154     port@1 {
155         reg = <1>;
156         dsi_out_panel: endpoint {
157             remote-endpoint = <&panel_in_dsi>;
158         };
159     };
160 };
161 };
162 };
163 &dsi0_in_vp2 {
164     status = "disabled";
165 };
166
167 &dsi0_in_vp3 {
168     status = "okay";
169 };
170
171 &route_dsi0 {
172     status = "okay";
173     connect = <&vp3_out_dsi0>;
174 };
175
176 &dsi0_panel {
177     power-supply = <&vcc5v0>;
178 };
179
180 &mipi_dcphy0 {
181     status = "okay";
182 };
183

```

关闭 LVDS

```
1 ▾ &dsi0 {  
2     status = "disabled";  
3     ...  
4 }  
5  
6 ▾ &dsi0_in_vp3 {  
7     status = "disabled";  
8 };  
9  
10 ▾ &route_dsi0 {  
11     status = "disabled";  
12 };
```

把这些状态修改为disabled即为关闭LVDS显示。

5.4.RTC

IDO-EVB3588-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/arm64/boot/dts/ido-evb3588s-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
```

5.5.UART

DTS配置

```
1 &uart6 {
2     status = "okay";
3     pinctrl-names = "default";
4     pinctrl-0 = <&uart6m1_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/ttyS6
```

收发测试

使用microcom可以进行串口收发测试，命令如下：

```
▼ Bash |
1  root@rk3588:~# microcom -s 115200 -p /dev/ttyS3
```

注意：测试完成，按Ctrl+x退出。