RP-RV1126 快速启动编译和使用说明

本文档主要介绍如何配置 RV1126 快速启动。

版本说明

日期	修订版本	修订内容	修改人	核定人	当前版本
2021-06-22	V1.0	初始版本	lixin	lixin	
2021-08-16	V1.1	1.针对 V2.1 版本优化文档内容 2.新增 LVGL8 支持说明	lixin	lixin	√

编译烧写说明

配置介绍

RV1126_rv1109 的快速启动的配置有 **BoardConfig-tb-v13.mk、BoardConfig-dualcam-tb-v13.mk、BoardConfig-battery-ipc.mk、BoardConfig-tb-v12.mk**，其中

1. BoardConfig-battery-ipc.mk 为电池产品，不适用于我司提供的开发板，也没做相关适配，且电源域并未使用rk809 管理，客户谨慎使用这个配置。

2. BoardConfig-dualcam-tb-v13.mk基于evb板的猫眼门锁产品，主要就是可以支持双目摄像头。

3. BoardConfig-tb-v13.mk 和 BoardConfig-tb-v12.mk 是基于evb板的快速启动ipc类产品，默认是支持单目摄像头。

当前版本的适配是基于 BoardConfig-tb-v13.mk 的配置做修改和增加的，在原来的基础上适配了我司的开发板硬件。

配置选择

执行 ./build.sh lunch 选择 BoardConfig-tb-v13.mk 对应的序号

```
$ ./build.sh lunch
processing option: lunch

You're building on Linux
Lunch menu...pick a combo:

0. default BoardConfig.mk
1. BoardConfig-38x38-spi-nand-ab.mk
2. BoardConfig-38x38-spi-nand.mk
3. BoardConfig-ab-v13.mk
4. BoardConfig-battery-ipc.mk
5. BoardConfig-dualcam-tb-v13.mk
6. BoardConfig-facial_gate.mk
```

```
7. BoardConfig-ramboot-uvc.mk
8. BoardConfig-robot.mk
9. BoardConfig-sl.mk
10. BoardConfig-slc-nand-v12.mk
11. BoardConfig-spi-nand.mk
12. BoardConfig-spi-nor-tb-v13.mk
13. BoardConfig-spi-nor-v12.mk
14. BoardConfig-tb-v12.mk
15. BoardConfig-tb-v13.mk
16. BoardConfig-uvcc-spi-nand.mk
17. BoardConfig-uvcc.mk
18. BoardConfig-v10-v11.mk
19. BoardConfig-v12.mk
20. BoardConfig.mk
21. pro-rv1109.mk
22. pro-rv1126.mk
23. rp-rv1109.mk
24. rp-rv1126.mk
Which would you like? [0]: 15  # 选择 BoardConfig-tb-v13.mk 对应的序号
```

说明： 不通的SDK版本序号可能会改变，只需选择 BoardConfig-tb-v13.mk 对应的序号即可

内核配置

此版型配置使用的内核 dts 文件为 `kernel/arch/arm/boot/dts/rv1126-evb-ddr3-v13-tb-emmc.dts` ，同样可以在此文件中配置 lcd 和 sensor。

```
/dts-v1/;
#include "rv1126.dtsi"
// #include "rv1126-evb-v13.dtsi"
#include "rongpin/rv1126_1109_common.dtsi"
#include "rv1126-thunder-boot-emmc.dtsi"
#include "rv1126-evb-thunder-boot.dtsi"

// -------------lcd  select ----------------
// #include "rongpin/rp_lcd_mipi_7inch_1024x600.dtsi"
// #include "rongpin/rp_lcd_mipi_5.5inch_720x1280.dtsi"
// #include "rongpin/rp_lcd_mipi_5.5inch_new_720x1280.dtsi"
// #include "rongpin/rp_lcd_mipi_8inch_800x1280.dtsi"
// #include "rongpin/rp_lcd_mipi_7inch_800x1280.dtsi"
#include "rongpin/rp_lcd_mipi_10inch_800x1280.dtsi"
// #include "rongpin/rp_lcd_mipi_10inch_1920x1200.dtsi"

// -------------camera select---------------
// #include "rongpin/camera-gc2053+ov2718.dtsi"
// #include "rongpin/camera-imx307x1.dtsi"
#include "rongpin/camera-gc2093x2.dtsi"
```

内核 menuconfig 使用的配置文件是 `kernel/arch/arm/configs/rv1126-tb.config` ，可以看到里面的很多功能都被配置成了 m，以ko的形式加载，如果是修改了内核的 menuconfig 配置，需要将修改的配置保存到 rv1126-tb.config 文件中，否则编译时配置会被撤销。

文件系统配置

文件系统的配置文件为 `buildroot/configs/rockchip_rv1126_evb_tb_defconfig`，文件系统也做了大量的删减，如需新增配置按一下步骤操作，否则可以直接前往 **编译**。

```
$ source envsetup.sh rockchip_rv1126_evb_tb
```

或者 source envsetup.sh 然后选择 rockchip_rv1126_evb_tb 对应的序号

```
$ source envsetup.sh
...
73. rockchip_rv1126_battery_ipc_nn
74. rockchip_rv1126_evb_dualcam_tb
75. rockchip_rv1126_evb_tb
76. rockchip_rv1126_robot
77. rockchip_rv1126_robot_recovery
78. rockchip_rv1126_rv1109
79. rockchip_rv1126_rv1109_ab
...
Which would you like? [0]: 75 # 选择 rockchip_rv1126_evb_tb 对应的序号
```

说明： 不通的SDK版本序号可能会改变，只需选择 rockchip_rv1126_evb_tb 对应的序号即可

然后执行 make menuconfig 后进入配置菜单

```
$ make menuconfig
```

配置完成后保存为默认配置

```
$ make savedefconfig
```

编译

配置完成后直接执行 ./build.sh 指令，即可编译所有镜像，需要注意的是，文件系统没有单独的编译选项，只能执行 ./build.sh 指令编译所有镜像的时候编译文件系统。

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

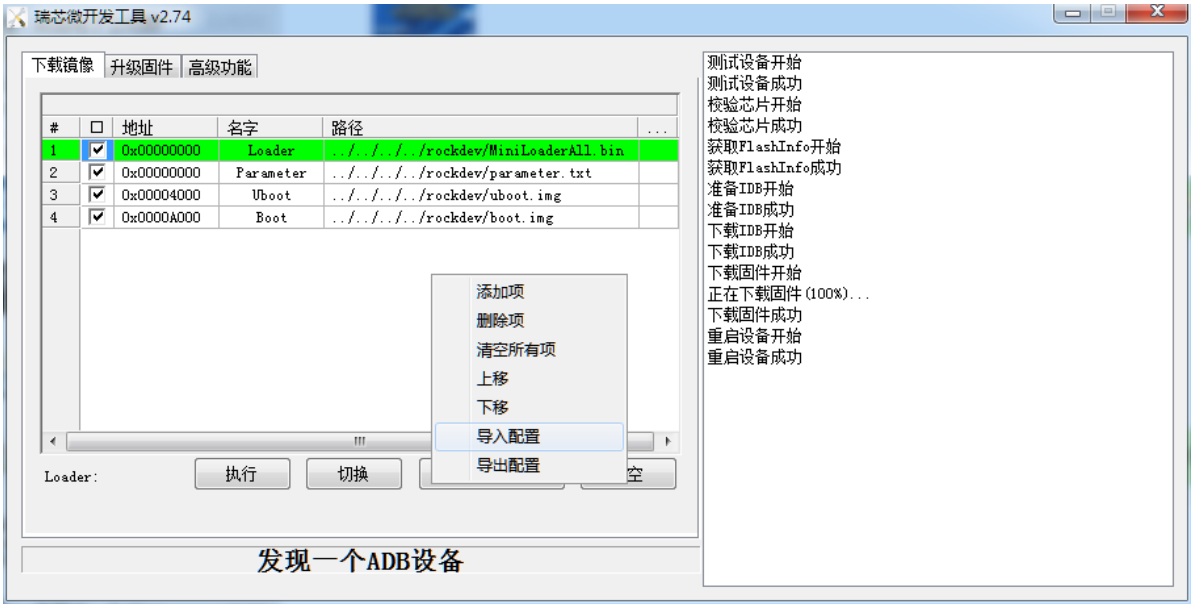
编译完成后同样会在 rockdev 目录下产生镜像文件

```
rockdev/
|-- MiniLoaderAll.bin -> ../u-boot/rv1126_spl_loader_v1.04.112.bin
|-- boot.img -> ../buildroot/output/rockchip_rv1126_evb_tb/images/ramboot.img
|-- misc.img -> ../device/rockchip/rockimg/wipe_all-misc.img
|-- parameter.txt -> ../device/rockchip/rv1126_rv1109/parameter-tb.txt
|-- uboot.img -> ../u-boot/uboot.img
|-- update.img
`-- userdata.img
```

烧写

烧写 update.img 操作请参考网盘链接根目录下的指导文档的烧写章节。

按分区烧写需要先导入配置才能按分区烧写



然后选择烧写工具目录下的 rv1126_rv1109_tb-config.cfg 文件。

调试开发说明

开启内核日志

默认不打印内核的任何日志信息，如果需要打开，可以修改 `kernel/arch/arm/boot/dts/rv1126-evb-ddr3-v13-tb-emmc.dts` 文件配置

```
chosen {
        bootargs = "loglevel=0 ..."
    };
```

修改 loglevel=7 打印 info 信息，修改 loglevel=8 将打印 debug 信息，然后重新编译烧写。

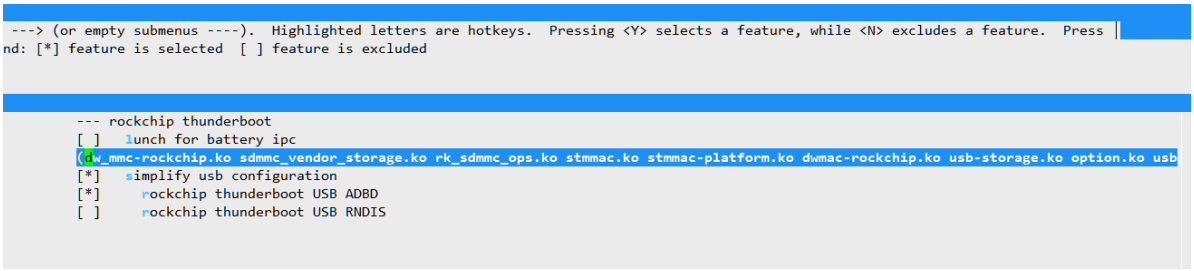
加载内核驱动说明

为了加快系统的启动速度，内核中将很多驱动都编译成了ko，在系统启动后再进行加载，如果需要将指定的ko 编译到文件系统中，待系统起来后加载，可按以下方式配置和添加。

1.sdk 根目录下执行指令打开 buildroot 系统的配置菜单

```
$ source envsetup.sh rockchip_rv1126_evb_tb && make menuconfig
```

然后按 / 搜索 thunderboot , 选中搜索结果进去后会看到以下界面



如图所示，如果是内核里面的ko文件，可以在这里添加 ko 的名称，无需指定路径，然后保存退出后保存为默认配置

```
$ make savedefconfig
```

2.然后重新编译 thunderboot ，这步是必须的，如果直接执行 ./build.sh 是不会将修改编译进去的

```
$ make thunderboot-dirclean && make thunderboot-rebuild
```

3.然后编辑 buildroot/board/rockchip/rv1126_rv1109/fs-overlay-evb-tb/etc/preinit.d/S07Load-module 文件，配置系统启动时 ko 文件的加载顺序和依赖的模块，

```
# 如配置加载以太网的 ko 顺序
ethernet ()
{
    echo "Enable ethernet function!" > /dev/kmsg
    insmod $MODULE_PATH/stmmac.ko
    insmod $MODULE_PATH/stmmac-platform.ko
    insmod $MODULE_PATH/dwmac-rockchip.ko
    ifconfig eth0 up
}

lunch_start()
{
    # 然后入口函数里面调用
    ethernet &
}
```

ko 的依赖可以在主机上使用 modeinfo 指令查看

```
$ modinfo dwmac-rockchip.ko | grep depends
depends:        stmmac-platform,stmmac
```

在 S07Load-module 脚本中，我司默认添加了一些功能，客户可以根据需求关闭或者打开。

```
lunch_start()
{
    ethernet &    # 以太网功能
    sdcard &      # sd卡，只能识别不能自动挂载
    usb_protocol  # USB协议驱动，如果需要使用USB功能，请先调用此函数
    ec20_4G &     # 4G
    udisk &       # U盘识别，只能识别不能自动挂载
    usbhid &      # 键鼠识别
    wifi &        # wifi功能
    # bluetooth & # 蓝牙功能
}
```

4.最后直接执行 ./build.sh 编译烧写就可以了。

文件系统增大注意

如果用户配置了其他的一些应用进去，导致文件系统过大的话，需要同步修改内核的配置，

编辑 kernel/arch/arm/boot/dts/rv1126-thunder-boot.dtsi 文件

```
ramdisk_r: ramdisk@2800000 {
    reg = <0x02800000 (48 * 0x00100000)>; // 这里是解压后的文件系统的大小. 48M
};

ramdisk_c: ramdisk@5800000 {
    reg = <0x05800000 (20 * 0x00100000)>; // 这里是压缩后的文件系统的大小, 20M
};

rkisp_thunderboot: rkisp@08000000 {
    reg = <0x08000000 (144 * 0x00100000)>;
};
```

需要确保解压前后的文件系统的大小要在数值范围内，否则会出现烧写后无法正常启动问题，压缩前后的文件系统大小可以使用如下指令获取

```
ls -l buildroot/output/rockchip_rv1126_evb_tb/images/
-rw-rw-r-- 1 ww ww 21100544 Jul  7 19:47 ramboot.img
-rw-r--r-- 1 ww ww 41860096 Jul  7 19:47 rootfs.romfs     # 压缩前的大小
-rw-rw-r-- 1 ww ww 16487643 Jul  7 19:47 rootfs.romfs.gz  # 压缩后的大小
-rw-r--r-- 1 ww ww 42301440 Jul  7 19:47 rootfs.tar
```

总内存比正常的少

快速启动的配置编译出来的镜像的内存会比正常少，是因为有部分预留给了SOC内部的MCU，MCU用做协助初始化ISP/Camera 和上电快速抓拍，当前版本暂未适配抓拍。

预留给MCU的内存 kernel/arch/arm/boot/dts/rv1126-thunder-boot.dtsi

```
memory: memory {
    device_type = "memory";
    reg = <0x00000000 0x20000000>;
};
```

这个部分的配置暂时不能删除，也不能修改，否则会影响系统启动。

rkmedia 使用说明

系统启动默认使用的是 rkmedia_vo_scale_test 显示，默认编译了rkmedia，如果需要使用 rkmedia 的 demo，请注意需要关闭 rkmedia_vo_scale_test。

```
$ killall rkmedia_vo_scale_test
```

wifi连接说明

直接调用脚本连接wifi

```
$ tb_start_wifi.sh RP405 rp778899aa
```

需要注意的是 tb_start_wifi.sh 脚本里面在获取 ip 地址的时候使用的是 udhcpc

```
function getdhcp() {
:<<eof
    while true
    do
        IPADDR=`dhd_priv wl dhcpc_dump | awk '{print $5}' | sed -n '3p'`
        if [ "$IPADDR" != "0.0.0.0" ];then
            NETMASK=`dhd_priv wl dhcpc_dump | awk '{print $7}' | sed -n '3p'`
            GW=`dhd_priv wl dhcpc_dump | awk '{print $9}' | sed -n '3p'`
            DNS=`dhd_priv wl dhcpc_dump | awk '{print $11}' | sed -n '3p'`

            ifconfig wlan0 $IPADDR netmask $NETMASK
            route add default gw $GW
            echo "nameserver $DNS" > /etc/resolv.conf

            echo $IPADDR $NETMASK $GW $DNS
            break
        fi
    done
eof

udhcpc -i wlan0 # 这里自动获取 ip 地址

}
```

MINIGUI 支持

TODO

LVGL支持

支持 LVGL8 ，相关demo可以到 sdk/app/lvgl 目录下查看，如果移动了文件夹位置，则需要重新配置编译工具路径和 drm 库的路径

```
# config gcc libdrm at first
GCC_COMPILER = $(realpath ../../prebuilts/gcc/linux-x86/arm/gcc-arm-8.3-2019.03-
x86_64-arm-linux-gnueabihf/)
DRM_SOURCE_DIR = drm
```

默认执行 make 即可编译 demo

```
make
```

编译完成生成一个名为 lvgl_demo 的可执行文件，然后直接将可执行文件复制到开发板运行

```
./lvgl_demo
```

开发参考：

官方网站： LVGL - Light and Versatile Embedded Graphics Library

官方github:： LVGL (github.com)

官方wiki： Welcome to the documentation of LVGL! — LVGL documentation

版本遗留问题

otg口无法切换至host模式

当前版本的 otg 口无法通过控制 gpio 的状态切换 host/otg 模式

调试串口开始打印乱码

调试串口一开始的时候会打印乱码，这个是因为在bootloader阶段的波特率为1500000，而且uboot和内核阶段的波特率为115200，波特率不一致导致的问题，这个不会影响功能。