T113 Linux

配置指南

版本号: 1.6

发布日期: 2022.05.31

版本历史

版本号	日期	制/修订人	内容描述
1.0	2021.06.23	allwinner	添加初版
1.1	2021.07.5	KPA0526	添加 SD 卡（sdc0）在 SPI NAND 上面的配置方法
1.2	2021.07.9	KPA0556	卡量产显示进度条配置
1.3	2021.07.12	KPA0526	添加 128G 存储卡测试方法
1.31	2021.08.31	KPA0526	添加 SPI NOR 常见问题
1.4	2021.09.10	KPA0526	1. 修改 SD 卡部分为通用性描述 2. 添加 LVDS、RGB LCD、CTP、以太网、OTA 配置方法 3. 添加 FAQ 4. 添加各模块硬件冲突说明
1.41	2021.09.29	KPA0526	添加调试串口的 sys_config 配置说明
1.42	2022.04.08	KPA0562	1. 添加 can 配置使用方法 2. 添加 standby 测试方法
1.5	2022.05.26	KPA0501	1. 增加 T113-i 的测试方法 2. 调整文档目录结构
1.6	2022.05.31	KPA0568	1. 修改 T113-i 以太网 IO 冲突 2. 增加 T113-i 的 wifi&bt 调试方法 3. 增加 T113-i 的 RS485 调试方法

目　录

插　图

1 概述

1.1 编写目的

介绍 T113 Linux 各功能配置方法和使用注意事项。

1.2 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	QT 版本
T113-S3	Linux-5.4	5.12.5
T113-i	Linux-5.4	5.12.5

说明

本文档涉及到硬件部分的描述，如未特别说明，均基于 *AllWinner T113-S3 EVB* 开发板，板上丝印 *“T113_EVB_V1_0”*。

说明

本文档涉及到 *dts, defconfig* 等配置文件路径的，对于方案、板型不同时，可能会有变化，具体根据自己的实际情况决定，*SDK* 根目录下.*buildconfig* 文件中会有当前 *defconfig* 的配置。

1.3 相关人员

T113 开发人员和 BSP 维护人员。

2 模块配置和使用

2.1 RTP 配置与使用

首先需要在t113\device\config\chips\t113\configs\evb1_auto_nand\board.dts中把 rtp 配成 okay：

```
&rtp {
    allwinner,tp-sensitive-adjust = <0xf>;
    allwinner,filter-type = <0x1>;
    allwinner,ts-attached;
    status = "okay";
};
```

内核方面，需要把 CONFIG_TOUCHSCREEN_SUNXI 选上，操作方法是：

```
cd out/kernel/build
make ARCH=arm menuconfig
选上 CONFIG_TOUCHSCREEN_SUNXI
```

驱动代码 kernel\linux-5.4\drivers\input\touchscreen\sunxi-ts.c

测试代码 platform\framework\auto\sdk_demo\tpadc

这个工具在 longan + emmc 方案默认会编译，但是在其他方案上不会默认编译。测试方法执行 tpadc_test，然后按触摸屏，串口会有坐标信息打印。

IO 冲突与硬件修改：RTP 为专用 IO 口，无 IO 复用，与其他模块不冲突，无需修改硬件。

2.2 CTP 配置与使用

T113-S3 板上 CTP 由于与 WiFi+BT, Camera, RGB888, I2S2, twi3 等冲突，在 dts 里面的状态默认是 disabled。需要自行修改 dts 开启。

文件路径device/config/chips/t113/configs/evb1_auto/linux-5.4/board.dts，修改如下：

```
diff --git a/configs/evb1_auto/linux-5.4/board.dts b/configs/evb1_auto/linux-5.4/board.dts
index 0b55f03..0c0d269 100644
--- a/configs/evb1_auto/linux-5.4/board.dts
+++ b/configs/evb1_auto/linux-5.4/board.dts
@@ -33,7 +33,7 @@
                regulator-min-microvolt = <5000000>;
                regulator-max-microvolt = <5000000>;
                regulator-enable-ramp-delay = <1000>;
```

```
-               gpio = <&pio PB 3 GPIO_ACTIVE_HIGH>;
+               /*gpio = <&pio PB 3 GPIO_ACTIVE_HIGH>;*/
                enable-active-high;
        };
 };
@@ -621,12 +621,12 @@
        twi_drv_used = <0>;
        dmas = <&dma 43>, <&dma 43>;
        dma-names = "tx", "rx";
-       status = "disabled";
+       status = "okay";
         ctp@14 {
                 compatible = "allwinner,goodix";
                 device_type = "ctp";
                 reg = <0x14>;
-                status = "disabled";
+                status = "okay";
                 ctp_name = "gt9xxnew_ts";
                 ctp_twi_id = <0x2>;
                 ctp_twi_addr = <0x14>;
@@ -667,7 +667,7 @@
        pinctrl-0 = <&twi3_pins_a>;
        pinctrl-1 = <&twi3_pins_b>;
        pinctrl-names = "default", "sleep";
-       status = "okay";
+       status = "disabled";
        ac108: ac108@3B {
                #sound-dai-cells = <0>;
                compatible = "Allwinner,MicArray_0";
@@ -885,7 +885,7 @@
 &vind0 {
        csi_top = <378000000>;
        csi_isp = <327000000>;
-       status = "okay"; /* enable twi and ver1 spi-nand max freq 30M */
+       status = "disabled";

        actuator0: actuator@5809450 {
                device_type = "actuator0";
@@ -1519,9 +1519,9 @@ tvd_row*tvd_column is the total tvd channel number to be used in
    multichannel mo
        usb_port_type = <0x2>;
        usb_detect_type = <0x1>;
        usb_detect_mode = <0>;
-       usb_id_gpio = <&pio PB 6 GPIO_ACTIVE_HIGH>;
+       /*usb_id_gpio = <&pio PB 6 GPIO_ACTIVE_HIGH>;*/
        enable-active-high;
-       usb_det_vbus_gpio = <&pio PB 2 GPIO_ACTIVE_HIGH>;
+       /*usb_det_vbus_gpio = <&pio PB 2 GPIO_ACTIVE_HIGH>;*/
        usb_wakeup_suspend = <0>;
        usb_serial_unique = <0>;
        usb_serial_number = "20080411";
```

以上修改内容分别是：注释 PB3 的使用，开启 twi0，开启 CTP，关闭 tiw3，关闭 Camera，注释 PB6 的使用，注释 PB2 的使用。

IO 冲突与硬件修改：

1.T113-S3 CTP 与 WiFi+BT, Camera, RGB888, I2S2, twi3 冲突，无法同时使用。硬件需

要断开 R293、R295。

2.T113-i CTP 使用 PE14-PE17 接口，与 GPHY-千兆网口有冲突。

2.3 按键测试

首先确保在t113\device\config\chips\t113\configs\evb1_auto_nand\board.dts 中把 gpadc 配成 okay：

```
&gpadc {
    channel_num = <2>;
    channel_select = <3>;
    channel_data_select = <3>;
    channel_compare_select = <3>;
    channel_cld_select = <3>;
    channel_chd_select = <3>;
    channel0_compare_lowdata = <1700000>;
    channel0_compare_higdata = <1200000>;
    channel1_compare_lowdata = <460000>;
    channel1_compare_higdata = <1200000>;
    status = "okay";
};
```

内核方面，保证 SUNXI_GPADC=y 宏已使能。

烧录成功后在串口中执行如下命令，然后按下开发板上的按键，就能有对应的打印输出。

```
T113-S3方案:
hexdump /dev/input/event2
T113-i方案:
hexdump /dev/input/event0
```

IO 冲突与硬件修改：按键使用 GPADC，GPADC 为专用 IO 口，无 IO 复用，与其他模块不冲突，IO 口默认已连接到按键，无需修改硬件。

2.4 USB 鼠标 & 键盘测试

```
cat /dev/input/eventX | hexdump
```

其中 X 为插入鼠标或键盘生成的 input 设备索引，如下打印则表示设备为 event5，即 X 为 5。

```
[  709.601759] input: Logitech Lenovo USB Optical Mouse as /devices/platform/soc@3000000
    /4200400.ohci1-controller/usb2/2-1/2-1:1.0/0003:17EF:6019.0001/input/input5
```

2.5 U 盘测试

board.dts 里把 usbc1 配成 okay，如下所示：

```
&usbc1 {
    device_type = "usbc1";
    usb_port_type = <0x01>;
    sb_detect_type = <0x1>;
    usb_regulator_io = "nocare";
    usb_wakeup_suspend = <0>;
    status = "okay";
};
```

插上 U 盘，对于 EMMC 板型，正常情况会自动挂载，并打印出挂载路径，如果没有自动挂载，执行如下命令手动挂载：

```
mkdir  /mnt/usb
mount /dev/sda  /mnt/usb/
ls /mnt/usb
```

usbc0 同理，只是默认情况下 usbc0 作为 adb 功能，若想把 U 盘接到 usbc0 上，需要把默认的 adb 功能禁止掉，方法如下：

```
cd /etc/init.d/
rm Sadbd
reboot
```

即把自启动的 Sadbd 删掉或者重命名，即可。

默认 U 盘挂载路径：`/mnt/usb/sda1/`

2.6 SD 卡测试

SD 卡座连接在 sdc0 接口，可用于外接存储卡存储文件、接量产卡用于卡量产、接启动卡用于卡启动。

在 T113-S3 EVB 板子上 sdc0（PF0~PF5）与 uart0（PF2、PF4）冲突，所以使用 SD 卡座时需要用其他 IO 作为调试串口，另外在 T113-S3 EVB 板上面除 uart0 外引出的是 uart3（PG8、PG9），SDK 默认是以 DVR 场景为基准，所以 EMMC 板型默认已经使能 SD 卡并且把调试串口配置在 uart3 上面。

由于 T113-S3 的 EMMC、SPI NAND 板型的默认调试串口配置是 uart3，并且默认已经使能 SD 卡，针对用户的不同需求，请选择性阅读本节内容：

- 需要 SD 卡，调试串口无所谓，能正常使用即可：EMMC 和 SPI NAND 板型无需修改，SPI NOR 板型参考 EMMC 板型的配置文件，参考uart0 的 IO 改为 PE2、PE3或调试串口配置成

uart2解决 IO 冲突问题，参考SDMMC 驱动使能方法，并配置 board.dts 在 sdc0 的 status 为 okay；

- 不需要 SD 卡，想将调试串口改回 uart0：参考调试串口从 uart3 改成 uart0将调试串口改回 uart0；
- 想使用 SD 卡同时使用 uart0 作为调试串口：参考调试串口从 uart3 改成 uart0将调试串口改回 uart0，参考uart0 的 IO 改为 PE2、PE3修改 uart0 复用到 PE2、PE3，此种情况适用于客户自己的 PCB；
- 想使用 SD 卡同时想使用其他串口作为调试串口：参考 EMMC 板型的配置文件，参考调试串口配置成 uart2的内容，修改调试串口到其他串口。
- T113-S3 使用 SD 卡时，PF2、PF4（原 uart0）的接口上面不能接串口线，避免 SD 卡读写异常！

T113-i EVB 板默认已正确配置 SD 卡（PF0~PF5），默认的调试串口为 uart0（PG17，PG18），故无需修改软硬件。

默认 SD 卡挂载路径：`/mnt/sdcard/mmcblk1p1/`

2.6.1 SDMMC 驱动使能方法

SPI NAND、SPI NOR 等板型的内核默认可能没有编译 SDMMC 驱动，需要手动修改内核 defconfig 打开相关配置。**不同板型内核 defconfig 可能不同，根据 SDK 根目录下面.buildconfig 文件决定**。

仓库路径：kernel/linux-5.4

```
diff --git a/arch/arm/configs/sun8iw20p1smp_fevb1_defconfig b/arch/arm/configs/
    sun8iw20p1smp_fevb1_defconfig
index e641c4d12e7c..3e4abd123b20 100644
--- a/arch/arm/configs/sun8iw20p1smp_fevb1_defconfig
+++ b/arch/arm/configs/sun8iw20p1smp_fevb1_defconfig
@@ -149,6 +149,8 @@ CONFIG_USB_SUNXI_USB_MANAGER=y
 CONFIG_USB_SUNXI_USB_DEBUG=y
 CONFIG_USB_SUNXI_USB_ADB=y
 CONFIG_USB_ROLE_SWITCH=y
+CONFIG_MMC=y
+CONFIG_MMC_SUNXI=y
 CONFIG_RTC_CLASS=y
 CONFIG_RTC_DRV_SUNXI=y
 CONFIG_DMADEVICES=y
```

改完后 build 一下，看看内核是否编译了 driver/mmc 下面的文件，编译了就说明配置成功了。

不确定的话看下 out/kernel/build/.config 这个文件，里面有如下内容则表示是正常的：

```
CONFIG_MMC=y
CONFIG_PWRSEQ_EMMC=y
CONFIG_PWRSEQ_SIMPLE=y
CONFIG_MMC_BLOCK=y
CONFIG_MMC_BLOCK_MINORS=8
```

```
CONFIG_MMC_SUNXI=y
CONFIG_MMC_SUNXI_V4P1X=y
CONFIG_MMC_SUNXI_V4P00X=y
CONFIG_MMC_SUNXI_V4P10X=y
CONFIG_MMC_SUNXI_V4P5X=y
CONFIG_MMC_SUNXI_V5P3X=y
```

如果没有这些，删掉 out/kernel/build/.config 重新编译，另外看下内核源码目录 kernel/linux-5.4 下面是否有.config，有的话也删掉。

说明

以下章节中多次涉及对 *uboot*、*kernel*、*buildroot* 的 *defconfig* 的修改，这些 *defconfig* 的定义都在 *SDK* 根目录下的*.buildconfig* 文件中，*LICHEE_BRANDY_DEFCONF* 指定了 *uboot* 的 *defconfig* 文件，*LICHEE_KERN_DEFCONF* 指定了 *kernel* 的 *defconfig* 文件，*LICHEE_BR_DEFCONF* 指定了 *buildroot* 的 *defconfig* 文件。修改 *defconfig* 文件后，需要重新进行*./build.sh config*，才会重新生成对应的配置文件。

2.7 128G 外部存储设备支持

由于内核空间的 NTFS 只支持读，不支持写，即使配置了 RW 权限也不行。所以将 NTFS、exFAT 文件系统的支持放到了 rootfs 去支持，也就是由 buildroot 支持。

对于 EMMC 板型，默认已经支持，并且支持自动挂载，无需修改。如需自行配置，参考如下内容。

2.7.1 buildroot 支持 NTFS、exFAT 文件系统

编译配置选择 longan 时，可在 buildroot 支持 NTFS、exFAT 文件系统，内核 defconfig 配置如下，删除了 exFAT 和 NTFS 支持，添加了 CONFIG_FUSE_FS 也就是用户空间文件系统支持：

```
diff --git a/arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig b/arch/arm/configs/
    sun8iw20p1smp_t113_auto_defconfig
index 12ccc04..e2b5773 100644
--- a/arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig
+++ b/arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig
@@ -3268,7 +3268,7 @@ CONFIG_ION_CMA_HEAP=y
 # CONFIG_FIELDBUS_DEV is not set
 # CONFIG_USB_WUSB_CBAF is not set
 # CONFIG_UWB is not set
-CONFIG_EXFAT_FS=y
+# CONFIG_EXFAT_FS is not set
 # CONFIG_GOLDFISH is not set
 # CONFIG_MFD_CROS_EC is not set
 # CONFIG_CHROME_PLATFORMS is not set
@@ -3528,7 +3528,7 @@ CONFIG_FILE_LOCKING=y
 # CONFIG_QUOTA is not set
 # CONFIG_AUTOFS4_FS is not set
 # CONFIG_AUTOFS_FS is not set
-# CONFIG_FUSE_FS is not set
```

```
+CONFIG_FUSE_FS=y
 # CONFIG_OVERLAY_FS is not set
 # CONFIG_INCREMENTAL_FS is not set

@@ -3554,9 +3554,9 @@ CONFIG_VFAT_FS=y
 CONFIG_FAT_DEFAULT_CODEPAGE=437
 CONFIG_FAT_DEFAULT_IOCHARSET="iso8859-1"
 # CONFIG_FAT_DEFAULT_UTF8 is not set
-CONFIG_NTFS_FS=y
+# CONFIG_NTFS_FS is not set
 # CONFIG_NTFS_DEBUG is not set
-CONFIG_NTFS_RW=y
+# CONFIG_NTFS_RW is not set
 # end of DOS/FAT/NT Filesystems

 #
```

buildroot 的 defconfig 修改如下：

```
diff --git a/configs/sun8iw20p1_t113_defconfig b/configs/sun8iw20p1_t113_defconfig
index ca46c64..9711a6a 100755
--- a/configs/sun8iw20p1_t113_defconfig
+++ b/configs/sun8iw20p1_t113_defconfig
@@ -964,8 +964,8 @@ BR2_PACKAGE_E2FSPROGS_FSCK=y
 # BR2_PACKAGE_E2FSPROGS_RESIZE2FS is not set
 # BR2_PACKAGE_E2TOOLS is not set
 # BR2_PACKAGE_ECRYPTFS_UTILS is not set
-# BR2_PACKAGE_EXFAT is not set
-# BR2_PACKAGE_EXFAT_UTILS is not set
+BR2_PACKAGE_EXFAT=y
+BR2_PACKAGE_EXFAT_UTILS=y
 # BR2_PACKAGE_F2FS_TOOLS is not set
 # BR2_PACKAGE_FLASHBENCH is not set
 # BR2_PACKAGE_FSCRYPTCTL is not set
```

修改后，可支持 exFAT 和 NTFS 的读写，并且挂载时无需指定文件系统。

2.7.2 内核支持 NTFS、exFAT 文件系统

如果是 BSP，没有 buildroot，只能在内核支持 exFAT 和 NTFS，exFAT 格式可以读写，NTFS 只能只读挂载。

内核 defconfig 的配置参考如下：

```
CONFIG_EXFAT_FS=y
CONFIG_NTFS_FS=y
CONFIG_NTFS_RW=y
```

支持 exFAT 和 NTFS 文件系统后，可使用 128G TF 卡，如果卡是做过启动卡或者量产卡，有多个分区，首先在 Windows 系统下使用 PhoenixCard 格式化，去掉多余分区。然后用 Windows 自带的格式化（文件管理器右键选择格式化），格式化为需要的文件系统，再插入开发板使用。

如果是内核支持的 exFAT，挂载时需要手动指定文件系统`mount -t exfat /dev/mmcblk0p1 /mnt`。

SDK 已支持 SD 卡热插拔功能，所以拔掉卡前需要手动umount /mnt。

内核支持的 exFAT 手动挂载操作如下：

```
[  193.997968] sunxi-mmc 4020000.sdmmc: failed to get HSSDR52_SDR25 used default
[  194.006154] mmc0: new high speed SDXC card at address aaaa
[  194.015651] mmcblk0: mmc0:aaaa SD128 119 GiB
[  194.026394]  mmcblk0: p1

/ # fdisk -l

Disk /dev/mmcblk0: 127.8 GB, 127865454592 bytes
255 heads, 63 sectors/track, 15545 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

        Device Boot      Start         End      Blocks  Id System
/dev/mmcblk0p1               3       15546   124852224   7 HPFS/NTFS
/ # mount -t exfat /dev/mmcblk0p1 /mnt
[  213.750672] [EXFAT] trying to mount...
[  213.850269] [EXFAT] mounted successfully
/ # cd /mnt/aaa/
/mnt/aaa # ls -al
total 512
drwxr-xr-x    2 0        0           131072 Feb  1  1980 .
drwxr-xr-x    4 0        0           131072 Jan  1 00:35 ..
-rwxr-xr-x    1 0        0               10 Feb  1  1980 1.c
-rwxr-xr-x    1 0        0               10 Sep 13  2021 bbb.txt
/mnt/aaa # cat bbb.txt
fsdfdfgd
/mnt/aaa # echo hello > hello.txt
/mnt/aaa # cat hello.txt
hello
/mnt/aaa #
```

如果读写时遇到 MMC 驱动报错，拔掉 uart0 的串口线，或检查硬件。

2.8 卡启动与卡量产测试

2.8.1 卡启动相关

T113-S3 evb1_auto 板在释放 PF2、PF4 并且使能 MMC 驱动后（即保证正常使用 SD 卡功能），卡量产功能就已经可以使用，即可使用 PhoenixCard 工具制作量产卡。

需要注意的是制作启动卡时，还需要保证固件配置中 sdc2 disable 并保持 sdc0 enable（status=okay），或者手动配置 env.cfg 里面的 bootcmd，否则卡启动时 rootfs 会变成 EMMC 里面的。原因是在 BROM 阶段，SOC 会去 try 多种启动介质，其中 SD 卡的优先级高于 EMMC，所以 boot0、uboot 都能正常从 SD 卡引导启动，但是到 kernel 阶段，如果没有 disable sdc2，内核将优先加载 EMMC，挂载到 mmcblk0 分区，然后加载 SD 卡，并挂载到 emmcblk1 分区，从而导致启动加载的 rootfs 实际上是 emmc 中的；而如果 EMMC 保持 disable 的状态，进入内核后 SD 卡依旧为 mmcblk0，所以就无需修改 env.cfg 里面的

bootcmd，也能保证 rootfs 是 SD 卡里面的。

uboot 相关代码如下（brandy/brandy-2.0/u-boot-2018/board/sunxi/board_helper.c）：

```
int sunxi_update_fdt_para_for_kernel(void)
{
        uint storage_type = 0;
.......省略部分代码
        /* fix nand&sdmmc */
        switch (storage_type) {
        case STORAGE_NAND:
.......省略部分代码
        case STORAGE_EMMC:
                fdt_enable_node("mmc2", 1);
                break;
        case STORAGE_EMMC0:
                fdt_enable_node("mmc0", 1);
                break;
        case STORAGE_EMMC3:
                fdt_enable_node("mmc3", 1);
                break;
        case STORAGE_SD:
                fdt_enable_node("mmc0", 1);
.......省略部分代码
        tick_printf("update dts\n");
        return 0;
}
```

目前 EMMC 板型已经默认 disable sdc2，如无问题，无需修改。

2.8.2 卡量产显示升级进度条

EMMC 板型默认已经支持，如果需要自行配置，参考以下内容。

在 lichee/brandy-2.0/u-boot-2018/configs/sun8iw20p1_auto_defconfig 中增加如下的配置：

```
diff --git a/configs/sun8iw20p1_auto_defconfig b/configs/sun8iw20p1_auto_defconfig
index 99d40ba..e3afe9a 100644
--- a/configs/sun8iw20p1_auto_defconfig
+++ b/configs/sun8iw20p1_auto_defconfig
@@ -80,6 +80,7 @@ CONFIG_ANDROID_BOOT_IMAGE=y
 #sprite
 CONFIG_SUNXI_SPRITE=y
 CONFIG_SUNXI_SECURE_STORAGE=y
+CONFIG_SUNXI_SPRITE_CARTOON=y

 #secure feature
 CONFIG_SUNXI_SECURE_BOOT=y
```

2.8.2.1 程序调用说明

程序主函数路径：brandy\brandy-2.0\u-boot-2018\cmd\sunxi_sprite.c。

```
int do_sprite_test(cmd_tbl_t *cmdtp, int flag, int argc, char * const argv[])
```

会判断 boot 当前的工作模式，如果是 WORK_MODE_CARD_PRODUCT，则跳转到

路径：brandy\brandy-2.0\u-boot-2018\sprite\sprite_main.c。

```
int sunxi_card_sprite_main(int workmode, char *name)

...

//启动动画显示
#ifdef CONFIG_SUNXI_SPRITE_CARTOON
sprite_cartoon_create(processbar_direct);
#endif
```

调用了uint sprite_cartoon_create(int op)函数创建进度条。

2.9 调试串口配置

2.9.1 uart0 的 IO 改为 PE2、PE3

本节内容适用于客户自行设计的 PCB，不适用于 Allwinner T113-S3 EVB 开发板，可参考方法但不能生搬硬套配置内容。对于不同的板型，配置文件路径需要根据实际情况决定。

要将 uart0 的 IO 改为 PE2、PE3，对板级目录做如下修改（仓库路径：device/config/chips/t113）：

```
diff --git a/configs/evb1_auto/linux-5.4/board.dts b/configs/evb1_auto/linux-5.4/board.dts
index e889c37..4a26037 100755
--- a/configs/evb1_auto/linux-5.4/board.dts
+++ b/configs/evb1_auto/linux-5.4/board.dts
@@ -149,14 +149,14 @@
        };

        uart0_pins_a: uart0_pins@0 {  /* For EVB1 board */
-               pins = "PF2", "PF4";
+               pins = "PE2", "PE3";
                function = "uart0";
                drive-strength = <10>;
                bias-pull-up;
        };

        uart0_pins_b: uart0_pins@1 {  /* For EVB1 board */
-               pins = "PF2", "PF4";
+               pins = "PE2", "PE3";
                function = "gpio_in";
```

```
        };

diff --git a/configs/evb1_auto/sys_config.fex b/configs/evb1_auto/sys_config.fex
index 1c4537a..ec08644 100644
--- a/configs/evb1_auto/sys_config.fex
+++ b/configs/evb1_auto/sys_config.fex
@@ -209,8 +209,8 @@ twi_sda  = port:PC1<3><1><default><default>
 ;----------------------------------------------------------------------------------
 [uart_para]
 uart_debug_port = 0
-uart_debug_tx   = port:PF02<3><1><default><default>
-uart_debug_rx   = port:PF04<3><1><default><default>
+uart_debug_tx   = port:PE02<6><1><default><default>
+uart_debug_rx   = port:PE03<6><1><default><default>


 ;----------------------------------------------------------------------------------
```

修改 sys_config.fex 时，需要注意 IO 的复用选择配置。此处 PF2、PF4 复用 3 是 uart0，PE2、PE3 复用 3 是 uart2；**如使用其他串口，需要根据实际的复用选择来修改**，比如 PB6、PB7 作为 uart3 用于调试串口，查找 user manual 的 9.7.5.1 PB Configure Register0 知复用是 7，应写为：

```
uart_debug_port = 3
uart_debug_tx   = port:PB06<7><1><default><default>
uart_debug_rx   = port:PB07<7><1><default><default>
```

注意这里修改的是 evb1_auto/linux-5.4/board.dts，实际上 evb1_auto/board.dts 就是链接到 evb1_auto/linux-5.4/board.dts 的，在 device/config/chips/t113/configs/evb1_auto 这个目录使用命令`ls -al`就能看到。不同的板型配置根据实际情况来修改。

修改后编译打包，可以使用 PE2、PE3 来调试，软件上面还是使用 uart0。

2.9.2 调试串口从 uart3 改成 uart0

T113-S3 EMMC、SPI NAND 板型默认使用的调试串口为 uart3，如果不使用 SD 卡，可以将调试串口改回 uart0，方法如下。

修改 sys_config.fex（仓库路径：device/config/chips/t113）：

```
diff --git a/configs/evb1_auto/sys_config.fex b/configs/evb1_auto/sys_config.fex
index c146e8a..3aba0af 100755
--- a/configs/evb1_auto/sys_config.fex
+++ b/configs/evb1_auto/sys_config.fex
@@ -208,9 +208,9 @@ twi_sda  = port:PC1<3><1><default><default>
 ;uart_debug_rx  |Boot串口接收的GPIO配置     |
 ;----------------------------------------------------------------------------------
 [uart_para]
-uart_debug_port = 3
-uart_debug_tx   = port:PG08<5><1><default><default>
-uart_debug_rx   = port:PG09<5><1><default><default>
+uart_debug_port = 0
```

```
+uart_debug_tx    = port:PF02<3><1><default><default>
+uart_debug_rx    = port:PF04<3><1><default><default>


 ;-----------------------------------------------------------------------------
```

修改 env.cfg（仓库路径：device/config/chips/t113）：

其中 setargs_nand、setargs_nand_ubi、setargs_mmc 这三行的修改都是把 earlycon 改为 earlyprintk。

```
diff --git a/configs/evb1_auto/longan/env.cfg b/configs/evb1_auto/longan/env.cfg
index 7ce079f..de1ae22 100755
--- a/configs/evb1_auto/longan/env.cfg
+++ b/configs/evb1_auto/longan/env.cfg
@@ -1,8 +1,8 @@

 #kernel command arguments
-earlycon=uart8250,mmio32,0x05000000
+earlyprintk=sunxi-uart,0x02500000
 initcall_debug=0
-console=ttyS3,115200
+console=ttyS0,115200
 nand_root=ubi0_5
 mmc_root=/dev/mmcblk0p5
 mtd_name=sys
@@ -17,9 +17,9 @@ specialstr=
 keybox_list=widevine,ec_key,ec_cert1,ec_cert2,ec_cert3,rsa_key,rsa_cert1,rsa_cert2,
    rsa_cert3
 dsp0_partition=dsp0
 #set kernel cmdline if boot.img or recovery.img has no cmdline we will use this
-setargs_nand=setenv bootargs ubi.mtd=${mtd_name} earlycon=${earlycon} clk_ignore_unused
+setargs_nand=setenv bootargs ubi.mtd=${mtd_name} earlyprintk=${earlyprintk} clk_ignore_unused
    initcall_debug=${initcall_debug} console=${console} loglevel=${loglevel} root=${
    nand_root} rootfstype=${rootfstype} i
-setargs_nand_ubi=setenv bootargs ubi.mtd=${mtd_name} earlycon=${earlycon}
+setargs_nand_ubi=setenv bootargs ubi.mtd=${mtd_name} earlyprintk=${earlyprintk}
    clk_ignore_unused initcall_debug=${initcall_debug} console=${console} loglevel=${
    loglevel} root=${nand_root} rootfstype=${rootfstyp
-setargs_mmc=setenv  bootargs earlycon=${earlycon} clk_ignore_unused initcall_debug=${
+setargs_mmc=setenv  bootargs earlyprintk=${earlyprintk} clk_ignore_unused initcall_debug=${
    initcall_debug} console=${console} loglevel=${loglevel} root=${mmc_root}  init=${init}
    partitions=${partitions} cma=${cm
+setargs_nand=setenv bootargs ubi.mtd=${mtd_name} earlyprintk=${earlyprintk}
    clk_ignore_unused initcall_debug=${initcall_debug} console=${console} loglevel=${
    loglevel} root=${nand_root} rootfstype=${rootfst
+setargs_nand_ubi=setenv bootargs ubi.mtd=${mtd_name} earlyprintk=${earlyprintk}
    clk_ignore_unused initcall_debug=${initcall_debug} console=${console} loglevel=${
    loglevel} root=${nand_root} rootfstype=${roo
+setargs_mmc=setenv  bootargs earlyprintk=${earlyprintk} clk_ignore_unused initcall_debug=$
    {initcall_debug} console=${console} loglevel=${loglevel} root=${mmc_root}  init=${init}
     partitions=${partitions} cm
 #nand command syntax: sunxi_flash read address partition_name read_bytes
 #0x4007f800 = 0x40080000(kernel entry) - 0x800(boot.img header 2k)
 boot_dsp0=sunxi_flash read 43000000 ${dsp0_partition};bootr 43000000 0 0
```

修改 board.dts（仓库路径：device/config/chips/t113）：

```
diff --git a/configs/evb1_auto/linux-5.4/board.dts b/configs/evb1_auto/linux-5.4/board.dts
index 0b55f03..2db4bb7 100644
--- a/configs/evb1_auto/linux-5.4/board.dts
```

```
+++ b/configs/evb1_auto/linux-5.4/board.dts
@@ -520,7 +520,7 @@
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&uart0_pins_a>;
        pinctrl-1 = <&uart0_pins_b>;
-       status = "disabled";
+       status = "okay";
 };

 &uart1 {
@@ -541,7 +541,7 @@
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&uart3_pins_a>;
        pinctrl-1 = <&uart3_pins_b>;
-       status = "okay";
+       status = "disabled";
 };

 &sdc2 {
```

修改 uboot 的 defconfig，删除以下三行内容：

```
CONFIG_BAUDRATE=115200
CONFIG_SPECIFY_CONSOLE_INDEX=y
CONFIG_CONS_INDEX=4
```

如果是 SPI NAND 板型，rootfs 的调试串口由 buildroot 决定，还需要修改 buildroot 的 defconfig 中如下配置：

```
BR2_TARGET_GENERIC_GETTY_PORT="ttyS3"
改为:
BR2_TARGET_GENERIC_GETTY_PORT="ttyS0"
```

2.9.3 调试串口配置成 uart2

本节内容适用于客户自行设计的 PCB，不适用于 Allwinner T113 EVB 开发板，可参考方法但不能生搬硬套配置内容。对于不同的板型，配置文件路径需要根据实际情况决定。

（1）修改 dts 使能 uart2

仓库路径：device/config/chips/t113。

修改如下：

```
diff --git a/configs/evb1_auto_nand/linux-5.4/board.dts b/configs/evb1_auto_nand/linux-5.4/
    board.dts
index 2579539..217299a 100755
--- a/configs/evb1_auto_nand/linux-5.4/board.dts
+++ b/configs/evb1_auto_nand/linux-5.4/board.dts
@@ -172,6 +172,18 @@
                function = "gpio_in";
        };
```

```
+       uart2_pins_a: uart2_pins@0 {
+               pins = "PE2", "PE3";
+               function = "uart2";
+               drive-strength = <10>;
+               bias-pull-up;
+       };
+
+       uart2_pins_b: uart2_pins@1 {
+               pins = "PE2", "PE3";
+               function = "gpio_in";
+       };
+
        twi0_pins_a: twi0@0 {
                pins = "PB10", "PB11";
                function = "twi0";
@@ -496,7 +508,14 @@
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&uart1_pins_a>;
        pinctrl-1 = <&uart1_pins_b>;
-       status = "disabled";
+       status = "okay";
+};
+
+&uart2 {
+       pinctrl-names = "default", "sleep";
+       pinctrl-0 = <&uart2_pins_a>;
+       pinctrl-1 = <&uart2_pins_b>;
+       status = "okay";
 };

 &sdc2 {
```

注意这里修改的是 evb1_auto_nand/linux-5.4/board.dts，实际上 evb1_auto_nand/board.dts 就是链接到 evb1_auto_nand/linux-5.4/board.dts 的，在 device/config/chips/t113/configs/evb1_auto_n 这个目录使用命令`ls -al`就能看到。

这里我们 enable 了 uart1、uart2，如果此时编译打包烧录，可以看到 ttyS0、ttyS1、ttyS2 都有了：

```
/ # ls /dev/ttyS*
/dev/ttyS0  /dev/ttyS1  /dev/ttyS2
```

（2）修改 env.cfg 使内核调试信息从 uart2 输出

仓库路径：device/config/chips/t113。

修改如下：

```
diff --git a/configs/evb1_auto_nand/bsp/env.cfg b/configs/evb1_auto_nand/bsp/env.cfg
index 48c4466..2393cc9 100755
--- a/configs/evb1_auto_nand/bsp/env.cfg
+++ b/configs/evb1_auto_nand/bsp/env.cfg
@@ -1,8 +1,9 @@

 #kernel command arguments
 earlyprintk=sunxi-uart,0x02500000
```

```
+earlycon=uart8250,mmio32,0x05000000
 initcall_debug=0
-console=ttyS0,115200
+console=ttyS2,115200
 nand_root=ubi0_5
 mmc_root=/dev/mmcblk0p5
 mtd_name=sys
@@ -17,9 +18,9 @@ specialstr=
 keybox_list=widevine,ec_key,ec_cert1,ec_cert2,ec_cert3,rsa_key,rsa_cert1,rsa_cert2,
    rsa_cert3
 dsp0_partition=dsp0
 #set kernel cmdline if boot.img or recovery.img has no cmdline we will use this
-setargs_nand=setenv bootargs ubi.mtd=${mtd_name} earlyprintk=${earlyprintk}
+setargs_nand=setenv bootargs ubi.mtd=${mtd_name} earlycon=${earlycon} clk_ignore_unused
    initcall_debug=${initcall_debug} console=${console} loglevel=${
    loglevel} root=${nand_root} rootfstype=${rootfstype} init=${init} partitions=${
    partitions} cma=${cma} snum=${snum} mac_addr=${mac} wifi_mac=${wifi_mac} bt_mac=${
    bt_mac} specialstr=${specialstr} gpt=1
-setargs_nand_ubi=setenv bootargs ubi.mtd=${mtd_name} earlyprintk=${earlyprintk}
+setargs_nand_ubi=setenv bootargs ubi.mtd=${mtd_name} earlycon=${earlycon}
    clk_ignore_unused initcall_debug=${initcall_debug} console=${console} loglevel=${
    loglevel} root=${nand_root} rootfstype=${rootfstype} init=${init} partitions=${
    partitions} cma=${cma} snum=${snum} mac_addr=${mac} wifi_mac=${wifi_mac} bt_mac=${
    bt_mac} specialstr=${specialstr} gpt=1
-setargs_mmc=setenv  bootargs earlyprintk=${earlyprintk} clk_ignore_unused initcall_debug=$
-    {initcall_debug} console=${console} loglevel=${loglevel} root=${mmc_root}  init=${init}
-     partitions=${partitions} cma=${cma} snum=${snum} mac_addr=${mac} wifi_mac=${wifi_mac}
-    bt_mac=${bt_mac} specialstr=${specialstr} gpt=1
+setargs_nand=setenv bootargs ubi.mtd=${mtd_name} earlycon=${earlycon} clk_ignore_unused
+    initcall_debug=${initcall_debug} console=${console} loglevel=${loglevel} root=${
+    nand_root} rootfstype=${rootfstype} init=${init} partitions=${partitions} cma=${cma}
+    snum=${snum} mac_addr=${mac} wifi_mac=${wifi_mac} bt_mac=${bt_mac} specialstr=${
+    specialstr} gpt=1
+setargs_nand_ubi=setenv bootargs ubi.mtd=${mtd_name} earlycon=${earlycon}
+    clk_ignore_unused initcall_debug=${initcall_debug} console=${console} loglevel=${
+    loglevel} root=${nand_root} rootfstype=${rootfstype} init=${init} partitions=${
+    partitions} cma=${cma} snum=${snum} mac_addr=${mac} wifi_mac=${wifi_mac} bt_mac=${
+    bt_mac} specialstr=${specialstr} gpt=1
+setargs_mmc=setenv  bootargs earlycon=${earlycon} clk_ignore_unused initcall_debug=${
+    initcall_debug} console=${console} loglevel=${loglevel} root=${mmc_root}  init=${init}
+    partitions=${partitions} cma=${cma} snum=${snum} mac_addr=${mac} wifi_mac=${wifi_mac}
+    bt_mac=${bt_mac} specialstr=${specialstr} gpt=1
 #nand command syntax: sunxi_flash read address partition_name read_bytes
 #0x4007f800 = 0x40080000(kernel entry) - 0x800(boot.img header 2k)
 boot_dsp0=sunxi_flash read 43000000 ${dsp0_partition};bootr 43000000 0 0
```

也就是修改了 console 为 ttyS2，添加 earlycon 这一行，并且把三个`earlyprintk=${earlyprintk}`都改为了`earlycon=${earlycon}`。

注意这个 env.cfg 的路径，在板级目录下面还区分了 bsp 和 longan，需要确定自己改的是实际用到的那个，如果不确定，改完之后 pack 一下，看 out/pack_out/env.cfg 是否和自己改的一样。

此时如果打包烧录，现象是内核阶段打印全都在 uart2，之前的 boot0、uboot 打印都在 uart0。如果在 uart2 输入命令有回显且能正常执行，则跳过 inittab 的修改，下一步直接修改 sysconfig 和 uboot 的 defconfig 即可。如果在 uart2 输入命令有回显但不能执行，则按顺序进行下一步 inittab 的修改。

（3）修改 inittab 使得 rootfs 改到 uart2

这里的修改最终目的都是使得 rootfs 中/etc/inittab 文件中的调试串口与内核调试串口一致。

编译选择 BSP 的，需要修改 env.cfg 文件，修改 console 为 ttyS2，目的是使得 rootfs 里面/etc/inittab 里面变成 uart2。

编译脚本 build/mkcmd.sh 里面 mk_defroot 这个函数里面有一行`substitute_inittab ${ROOTFS}/etc/inittab`会用 env.cfg 里面的 console 设置去替换/etc/inittab 里面的设置，需要根据这里实际用的 env.cfg 去修改，可以在 build/mkcmd.sh 的 substitute_inittab 函数中打印出 env_cfg_dir，确定实际生效的那个。

参考修改如下，实际修改文件根据 substitute_inittab 函数选择的为准：

```
diff --git a/configs/default/env.cfg b/configs/default/env.cfg
index 30fe670..88dfab1 100644
--- a/configs/default/env.cfg
+++ b/configs/default/env.cfg
@@ -2,7 +2,7 @@
 #kernel command arguments
 earlyprintk=sunxi-uart,0x02500000
 initcall_debug=0
-console=ttyS0,115200
+console=ttyS2,115200
 nand_root=ubi0_5
 mmc_root=/dev/mmcblk0p5
 mtd_name=sys
```

这里的修改需要 build 才能生效。

确定修改是否生效，查看`out/t113/evb1_auto_nand/bsp/rootfs_def/etc/inittab`是否修改成功：

```
修改env.cfg前：
::respawn:/sbin/getty -n -L  ttyS0 115200 vt100 -n -l /bin/sh # GENERIC_SERIAL
修改env.cfg并且build后：
::respawn:/sbin/getty -n -L  ttyS2 115200 vt100 -n -l /bin/sh # GENERIC_SERIAL
```

如果选择的是 longan，会编译 buildroot，则/etc/inittab 由 buildroot 生成，并且由 platform/config 目录下的脚本做了二次修改，rootfs 的调试串口与内核一致，无需修改，/etc/inittab 部分内容如下：

```
# Put a getty on the serial port
#ttyS0::respawn:/sbin/getty -L  ttyS0 115200 vt100 # GENERIC_SERIAL
::respawn:-/bin/sh
```

这种情况下 rootfs 的调试串口与内核一致，当然也可以和 bsp 的一样，修改为 ttyS2，要修改为 ttyS2，做以下修改：

```
仓库路径platform/config
diff --git a/buildroot/post_build.sh b/buildroot/post_build.sh
index ca8b340..e7ac60b 100755
--- a/buildroot/post_build.sh
+++ b/buildroot/post_build.sh
```

```
@@ -15,9 +15,9 @@ add_preinit_to_inittab(){
                fi

                #commented ttyS0, insert /bin/sh
-               grep "#ttyS0::respawn:" ${TARGET_DIR}/etc/inittab >/dev/null
+               grep "ttyS2" ${TARGET_DIR}/etc/inittab >/dev/null
                if [ $? -eq 0 ]; then
-                       echo "ttyS0 is already commented!"
+                       echo "ttyS2 is already exist!"
                else
                        echo "ttyS0 is not commented, commented it!"
                        sed -i 's/ttyS0/#&/' ${TARGET_DIR}/etc/inittab
@@ -27,10 +27,10 @@ add_preinit_to_inittab(){
                                echo "/bin/sh is already exist!"
                        else
                                echo "insert ::respawn:-/bin/sh"
-                               sed -i '/GENERIC_SERIAL/a ::respawn:-/bin/sh' ${TARGET_DIR
    }/etc/inittab
+                               sed -i '/GENERIC_SERIAL/a ::respawn:/sbin/getty -n -L
    ttyS2 115200 vt100 -n -l /bin/sh # GENERIC_SERIAL' ${TARGET_DIR}/etc/inittab
                        fi
                fi
        fi
 }


仓库路径buildroot/buildroot-201902:
diff --git a/package/busybox/busybox.mk b/package/busybox/busybox.mk
index 5c5a8ea839..503593ec7e 100644
--- a/package/busybox/busybox.mk
+++ b/package/busybox/busybox.mk
@@ -214,7 +214,7 @@ endef

 ifeq ($(BR2_TARGET_GENERIC_GETTY),y)
 define BUSYBOX_SET_GETTY
-       $(SED) '/# GENERIC_SERIAL$$/s~^.*#~$(SYSTEM_GETTY_PORT)::respawn:/sbin/getty -L $(
    SYSTEM_GETTY_OPTIONS) $(SYSTEM_GETTY_PORT) $(SYSTEM_GETTY_BAUDRATE) $(SYSTEM_GETTY_TERM
    ) #~' \
+       $(SED) '/# GENERIC_SERIAL$$/s~^.*#~ttyS2::respawn:/sbin/getty -L $(
    SYSTEM_GETTY_OPTIONS) ttyS2 $(SYSTEM_GETTY_BAUDRATE) $(SYSTEM_GETTY_TERM) #~' \
                $(TARGET_DIR)/etc/inittab
 endef
 BUSYBOX_TARGET_FINALIZE_HOOKS += BUSYBOX_SET_GETTY
```

然后重新 build，/etc/inittab 部分内容如下：

```
# Put a getty on the serial port
ttyS2::respawn:/sbin/getty -L  ttyS2 115200 vt100 # GENERIC_SERIAL
```

如果以上修改无效，说明 buildroot 没有定义 post_build.sh 脚本，这时需要修改 buildroot 的 defconfig，按如下配置：

```
BR2_TARGET_GENERIC_GETTY_PORT="ttyS0"
改为:
BR2_TARGET_GENERIC_GETTY_PORT="ttyS2"
```

此时如果编译打包烧录，现象是内核阶段打印全都在 uart2，之前的 boot0、uboot 打印都在

uart0，在 uart2 可以正常输入命令并执行，在 uart0 输入无反应。

（4）修改 sysconfig 和 uboot 的 defconfig 使得 BootLoader 阶段改到 uart2

修改 sys_config.fex，仓库路径：device/config/chips/t113：

```
diff --git a/configs/evb1_auto_nand/sys_config.fex b/configs/evb1_auto_nand/sys_config.fex
index 3403ed4..680eac5 100755
--- a/configs/evb1_auto_nand/sys_config.fex
+++ b/configs/evb1_auto_nand/sys_config.fex
@@ -192,9 +192,9 @@ twi_sda  = port:PC1<3><1><default><default>
 ;uart_debug_rx  |Boot串口接收的GPIO配置     |
 ;-------------------------------------------------------------------------------
 [uart_para]
-uart_debug_port = 0
-uart_debug_tx   = port:PF02<3><1><default><default>
-uart_debug_rx   = port:PF04<3><1><default><default>
+uart_debug_port = 2
+uart_debug_tx   = port:PE02<3><1><default><default>
+uart_debug_rx   = port:PE03<3><1><default><default>


 ;-------------------------------------------------------------------------------
```

修改 sys_config.fex 时，需要注意 IO 的复用选择配置。此处 PF2、PF4 复用 3 是 uart0，PE2、PE3 复用 3 是 uart2；**如使用其他串口，需要根据实际的复用选择来修改**，比如 PB6、PB7 作为 uart3 用于调试串口，查找 user manual 的 9.7.5.1 PB Configure Register0 知复用是 7，应写为：

```
uart_debug_port = 3
uart_debug_tx   = port:PB06<7><1><default><default>
uart_debug_rx   = port:PB07<7><1><default><default>
```

修改 sys_config.fex 后，boot0 阶段打印应在修改后的串口上。

修改 uboot 的 defconfig，仓库路径：brandy/brandy-2.0/u-boot-2018（**这里的 defconfig 需要根据.buildconfig 里面 LICHEE_BRANDY_DEFCONF 这一项决定**）。

```
diff --git a/configs/sun8iw20p1_defconfig b/configs/sun8iw20p1_defconfig
index 3ecc35b32d..23a84b6542 100644
--- a/configs/sun8iw20p1_defconfig
+++ b/configs/sun8iw20p1_defconfig
@@ -109,6 +109,11 @@ CONFIG_CMD_SUNXI_BOOTR=y
 CONFIG_XTENSA_DSP=y


+# Serial drivers
+CONFIG_BAUDRATE=115200
+CONFIG_SPECIFY_CONSOLE_INDEX=y
+CONFIG_CONS_INDEX=3
+


 #
```

现在如果编译打包烧录，正常的话全部的打印都应该在 uart2 了（包括 USB 烧录的时候），并且 uart2 可以正常输入命令并执行，ADB 等其他功能正常运行不受影响。uart0 输入不再有反应。

（5）disable uart0

上面已经把调试串口改到了 uart2，现在我们要 disable uart0。

修改 board.dts 把 uart0、uart1 都 disable，这时候再读复用寄存器：

```
/sys/class/sunxi_dump # echo 0x020000f0 > dump
/sys/class/sunxi_dump # cat dump
0x0f3f3f33
```

PF2、PF4、PF6 都已经变成 disable 状态。

2.10 LVDS LCD 配置

T113 EMMC 和 SPI NAND 板型默认使用 LVDS 屏。默认情况无需修改软件，如需自行配置，参考以下内容：

以 T113-S3 为例，dts 需要修改两个文件，分别是`device/config/chips/t113/configs/evb1_auto/linux-5.4/board.dts`和`device/config/chips/t113/configs/evb1_auto/uboot-board.dts`，修改两个 dts 中 lcd0 的配置，并且修改 disp 节点中 fb0_width, fb0_height 的值使之与 LCD 的分辨率匹配。

lcd0 配置参考如下：

```
&lcd0 {
        lcd_used            = <1>;

        lcd_driver_name     = "default_lcd";
        lcd_backlight       = <50>;
        lcd_if              = <3>;

        lcd_x               = <1024>;
        lcd_y               = <600>;
        lcd_width           = <150>;
        lcd_height          = <94>;
        lcd_dclk_freq       = <52>;

        lcd_pwm_used        = <1>;
        lcd_pwm_ch          = <7>;
        lcd_pwm_freq        = <50000>;
        lcd_pwm_pol         = <1>;
        lcd_pwm_max_limit   = <255>;

        lcd_hbp             = <160>;
        lcd_ht              = <1344>;
        lcd_hspw            = <60>;
        lcd_vbp             = <23>;
        lcd_vt              = <638>;
        lcd_vspw            = <10>;
```

```
        lcd_lvds_if         = <0>;
        lcd_lvds_colordepth = <0>;
        lcd_lvds_mode       = <0>;
        lcd_frm             = <1>;
        lcd_hv_clk_phase    = <0>;
        lcd_hv_sync_polarity= <0>;
        lcd_gamma_en        = <0>;
        lcd_bright_curve_en = <0>;
        lcd_cmap_en         = <0>;

        deu_mode            = <0>;
        lcdgamma4iep        = <22>;
        smart_color         = <90>;

        pinctrl-0 = <&lvds0_pins_a>;
        pinctrl-1 = <&lvds0_pins_b>;
};
```

如果出现无背光，请检查背光电源。对于其他 LVDS 屏，需要自行调节屏参。

IO 冲突与硬件修改:

1.LVDS 与 MIPI DSI, RGB LCD 的引脚互相冲突，不能同时使用;

2.T113-S3 evb 板的 LVDS 默认已连接，无需修改硬件，排线需接到 J4 座子上面;

3.T113-i evb 板需要确保播码开关已关闭 SW1 和 SW2，避免 RGB 信号影响，同时排线需接到 J4 座子上面。

2.11 RGB LCD 配置

T113 EMMC 和 SPI NAND 板型默认使用 LVDS 屏。RGB 和 MIPI 屏的配置是注释状态的，需要自行修改 dts 文件取消注释 RGB 配置，并将 LVDS 的配置注释掉。

以 T113-S3 为例，dts 需要修改两个文件，分别是`device/config/chips/t113/configs/evb1_auto/linux-5.4/board.dts`和`device/config/chips/t113/configs/evb1_auto/uboot-board.dts`，修改两个 dts 中 lcd0 的配置，并且修改 disp 节点中 fb0_width, fb0_height 的值使之与 LCD 的分辨率匹配。

RGB 屏的 lcd0 配置参考如下:

```
&lcd0 {
        lcd_used            = <1>;

        lcd_driver_name     = "default_lcd";
        lcd_backlight       = <50>;
        lcd_if              = <0>;

        lcd_x               = <800>;
        lcd_y               = <480>;
        lcd_width           = <150>;
        lcd_height          = <94>;
```

```
        lcd_dclk_freq       = <48>;

        lcd_pwm_used        = <1>;
        lcd_pwm_ch          = <7>;
        lcd_pwm_freq        = <50000>;
        lcd_pwm_pol         = <1>;

        lcd_hbp             = <55>;
        lcd_ht              = <1240>;
        lcd_hspw            = <20>;
        lcd_vbp             = <35>;
        lcd_vt              = <650>;
        lcd_vspw            = <10>;

        lcd_lvds_if         = <0>;
        lcd_lvds_colordepth = <1>;
        lcd_lvds_mode       = <0>;
        lcd_frm             = <0>;
        lcd_io_phase        = <0x0000>;
        lcd_gamma_en        = <0>;
        lcd_bright_curve_en = <0>;
        lcd_cmap_en         = <0>;

        deu_mode            = <0>;
        lcdgamma4iep        = <22>;
        smart_color         = <90>;

        pinctrl-0 = <&rgb18_pins_a>;
        pinctrl-1 = <&rgb18_pins_b>;
};
```

如果出现无背光，请检查背光电源。对于其他 RGB 屏，需要自行调节屏参。

IO 冲突与硬件修改:

1.RGB LCD 与 MIPI DSI, LVDS 的引脚互相冲突，不能同时使用；

2.T113-S3 evb 板 RGB666/RGB888 默认已连接，无需修改硬件，排线需接到 J6 座子上面，另外由于 RGB888 用到了 PB2-PB7，故不能与 Camera、WiFi+BT、CTP、I2S2 同时使用；

3.T113-i evb 板 RGB666 屏可直接关闭 SW6，SW7, 打开 SW1，SW2，SW3，RGB888 屏由于使用到 PB2-PB7，还需要打开 SW5，即关闭 SW6，SW7, 打开 SW1，SW2，SW3，排线需接到 J6 座子上面；

2.12 MIPI-DSI

注意：MIPI-DSI 屏一定要用 40pin 座 DSI 转接板（转线序用，接口不变，插在 RGB 屏输出接口 J6 上面），如果不接会烧坏屏。

2.12.1 MIPI-DSI 显示配置

T113-S3 EMMC 板型默认已经配置了 RGB、LVDS、MIPI 三种屏幕，默认是 LVDS，需要使用 MIPI DSI 时，将 LVDS 那组参数注释掉，取消 MIPI LCD 的注释即可。

在 uboot-board.dts 和 board.dts 中，搜索 he0801a068，打开此 LCD 的配置，关闭 LVDS 的配置（两个 dts 都要配置）。

如果没有，可以按照如下配置添加：

```
&lcd0 {
        lcd_used            = <1>;

        lcd_driver_name     = "he0801a068";
        lcd_backlight       = <50>;
        lcd_if              = <4>;

        lcd_x               = <800>;
        lcd_y               = <1280>;
        lcd_width           = <800>;
        lcd_height          = <1280>;
        lcd_dclk_freq       = <78>;

        lcd_pwm_used        = <1>;
        lcd_pwm_ch          = <7>;
        lcd_pwm_freq        = <50000>;
        lcd_pwm_pol         = <1>;
        lcd_pwm_max_limit   = <255>;

        lcd_hbp             = <149>;
        lcd_ht              = <978>;
        lcd_hspw            = <16>;
        lcd_vbp             = <7>;
        lcd_vt              = <1329>;
        lcd_vspw            = <5>;

        lcd_dsi_lane        = <4>;
        lcd_dsi_if          = <0>;
        lcd_dsi_format      = <0>;
        lcd_dsi_te          = <0>;
        lcd_frm             = <0>;
        lcd_io_phase        = <0>;
        lcd_gamma_en        = <0>;
        lcd_bright_curve_en = <0>;
        lcd_cmap_en         = <0>;

        deu_mode            = <0>;
        lcdgamma4iep        = <22>;
        smart_color         = <90>;

        lcd_gpio_0 = <&pio PG 13 GPIO_ACTIVE_HIGH>;
        pinctrl-0 = <&dsi4lane_pins_a>;
        pinctrl-1 = <&dsi4lane_pins_b>;
};
```

在两个 dts 中设置 fb0_width、fb0_height，与屏幕匹配。

重新编译固件并烧入后，开机显示正常。

IO 冲突与硬件修改：

1.MIPI DSI 与 LVDS, RGB LCD 的引脚互相冲突，不能同时使用；

2.T113-S3 evb 板 MIPI_DSI 默认已连接，无需修改硬件，排线需接到 J6 座子上面；

3.T113-i evb 板播码开关需打开 SW1，SW2，SW3，并且去掉 R952，R953，R954，同时在 R971，R972，R973 接上 0 欧电阻，排线需接到 J6 座子上面；

2.13 LCD 背光测试

```
mount -t debugfs none /sys/kernel/debug;
cd /sys/kernel/debug/dispdbg;
echo lcd0 > name;echo setbl > command;echo 200 > param;echo 1 > start;
echo lcd0 > name;echo setbl > command;echo 255 > param;echo 1 > start;
```

使用背光 PWM 调节，需要确定 board.dts 中已经配置了 lcd_pwm 相关内容，并且与其他模块无 IO 冲突，并且硬件上面把 PWM 的 IO 正确连接到了背光 IC 的 EN 脚。

2.14 Camera 输入-CVBS IN 配置

在 kernel/linux-5.4/arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig 中增加如下的宏：

```
diff --git a/arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig b/arch/arm/configs/
    sun8iw20p1smp_t113_auto_defconfig
index 2c47398bb124..0ab6fb7b108f 100644
--- a/arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig
+++ b/arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig
@@ -2261,7 +2261,9 @@ CONFIG_MEDIA_SUPPORT=y
 #
 # Multimedia core support
 #
-# CONFIG_MEDIA_CAMERA_SUPPORT is not set
+CONFIG_MEDIA_CAMERA_SUPPORT=y
+CONFIG_V4L_PLATFORM_DRIVERS=y
+CONFIG_VIDEO_SUNXI_TVD=y
 # CONFIG_MEDIA_ANALOG_TV_SUPPORT is not set
 # CONFIG_MEDIA_DIGITAL_TV_SUPPORT is not set
 # CONFIG_MEDIA_RADIO_SUPPORT is not set
```

测试方式和其他 csi camera 的方式一样，通过 ls /dev/video* 查看具体的 video 节点，然后通过 sdkdemo 中的 csitest 做测试，具体操作效果如下：

```
# ./csitestT5 4 0 720 576 / 4 100 25
open /dev/video4 fd = 3[   63.413766] [tvd] vidioc_s_fmt_vid_cap:1598
[   63.413766] interface=0
[   63.413766] system=PAL
[   63.413766] format=0
[   63.413766] output_fmt=YUV420

[   63.431522] [tvd] vidioc_s_fmt_vid_cap:1602
[   63.431522] row=1
[   63.431522] column=1
[   63.431522] ch[0]=0
[   63.431522] ch[1]=0
[   63.431522] ch[2]=0
[   63.431522] ch[3]=0
[   63.450467] [tvd] vidioc_s_fmt_vid_cap:1604
[   63.450467] width=720
[   63.450467] height=576
[   63.450467] dev->sel=0
[   63.463318] [tvd] tvd_cagc_and_3d_config:1456 tvd0 agc auto mode
[   63.470268] [tvd] tvd_cagc_and_3d_config:1465 tvd0 CAGC enable:0x1
[   63.481394] [tvd] tvd_cagc_and_3d_config:1492 tvd0 3d enable :0x46100000
resolution got from sensor = 720*576 num_planes = 0
[   63.695862] [tvd] vidioc_streamon:1687 Out vidioc_streamon:0
VIDIOC_STREAMON ok
[   63.758290] [tvd] tvd_isr:785 In tvd_isr
VIDIOC_STREAMOFF ok
mode 4 test done at the 0 time!![   67.782622] [DISP] disp_mgr_set_layer_config,line:1597:

time cost 4.369185(s)
[   67.782628] NULL hdl!
# ls
THIS_IS_NOT_YOUR_ROOT_FILESYSTEM  mnt
bin                               opt
csitestT5                         proc
dev                               root
etc                               run
fb4_yuv4_720_576.bin              sbin
init                              sys
lib                               system
lib32                             tmp
linuxrc                           usr
lost+found                        var
media
```

图 2-1: 操作效果

2.15 Camera 输入-N5 配置

EMMC 板型默认已经支持 Camera，软件无需修改，如果需要自行配置，参考以下内容。

2.15.1 IO 冲突与硬件修改

camera 用到 IO 较多，其中 T113-S3 上与以太网、RGB888、I2S2、CTP、WiFi、BT 冲突，T113-i 上与以太网冲突。

T113-S3 evb 硬件需要修改，需要断开 R921, R922, R923, R924, R846, R848, R919, R920, R917, R918, R804, R805；连接 R925, R926, R927, R929, R932, R933, R276, R274, R341, R339, R858, R860。

T113-S3 硬件修改截图如下：

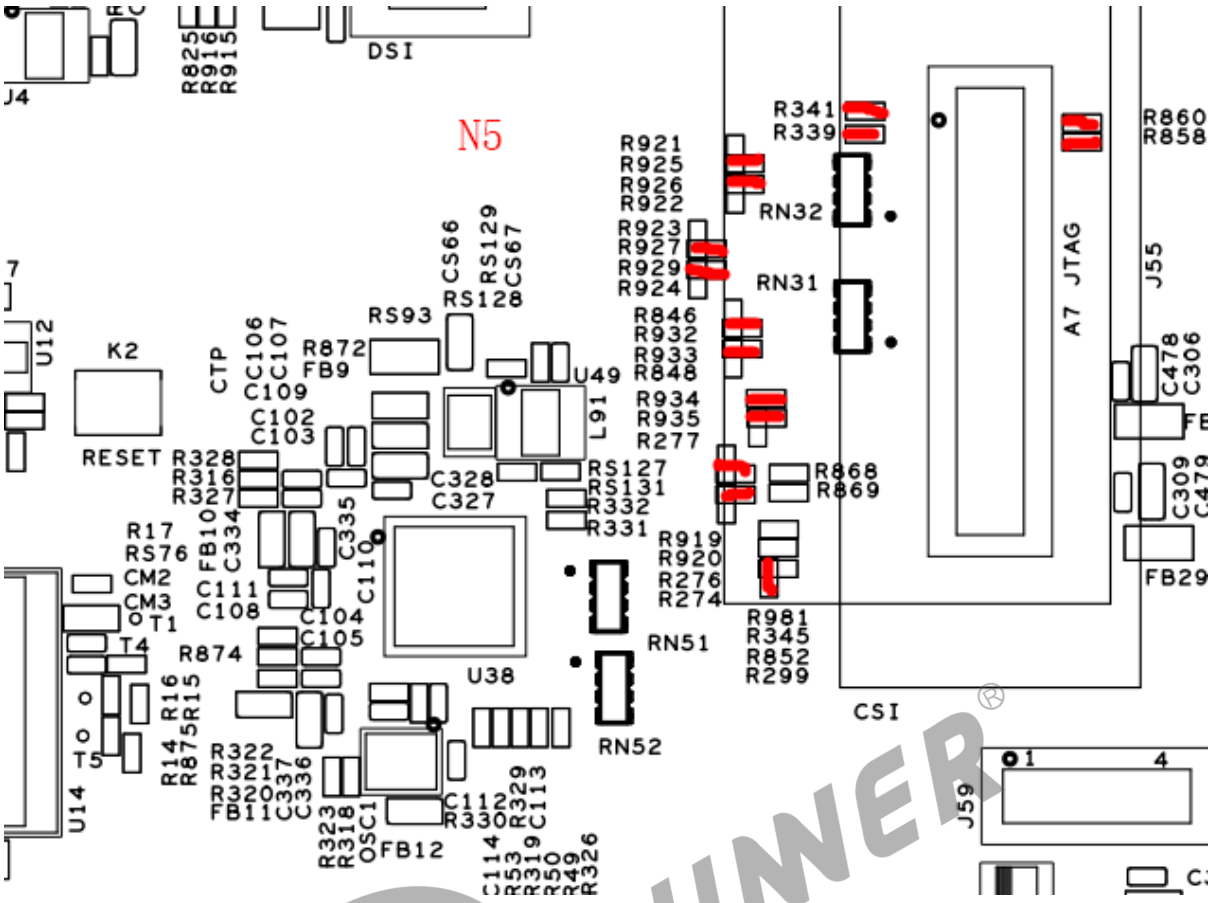

图 2-2: 红色表示要连接的器件

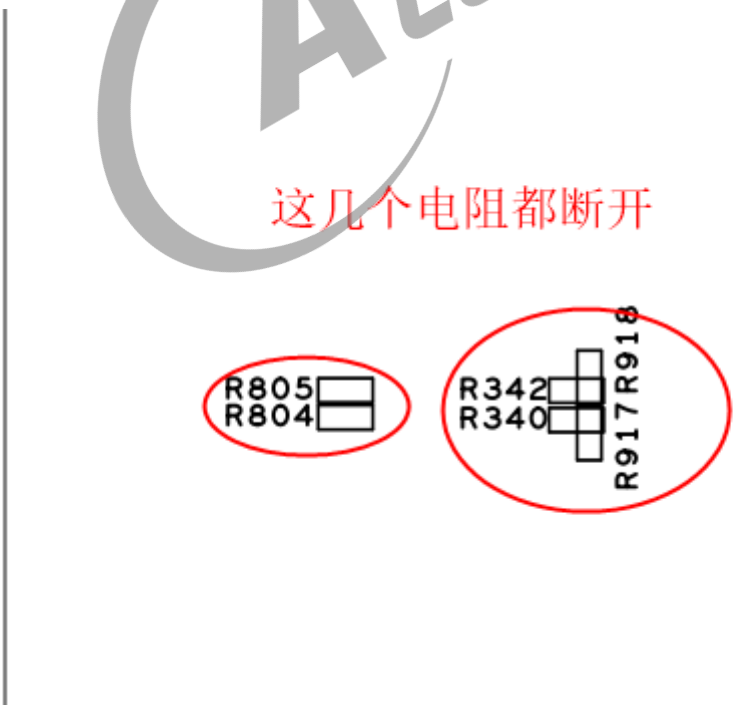

图 2-3: 需要断开的器件

T113-i evb 硬件只需要打开 SW8 和 SW9，关闭 SW10 和 SW11 即可使用 N5 功能。

2.15.2 软件修改 menuconfig 配置

执行如下命令进行配置：

```
./build.sh ARCH=arm menuconfig
```

具体修改如下：

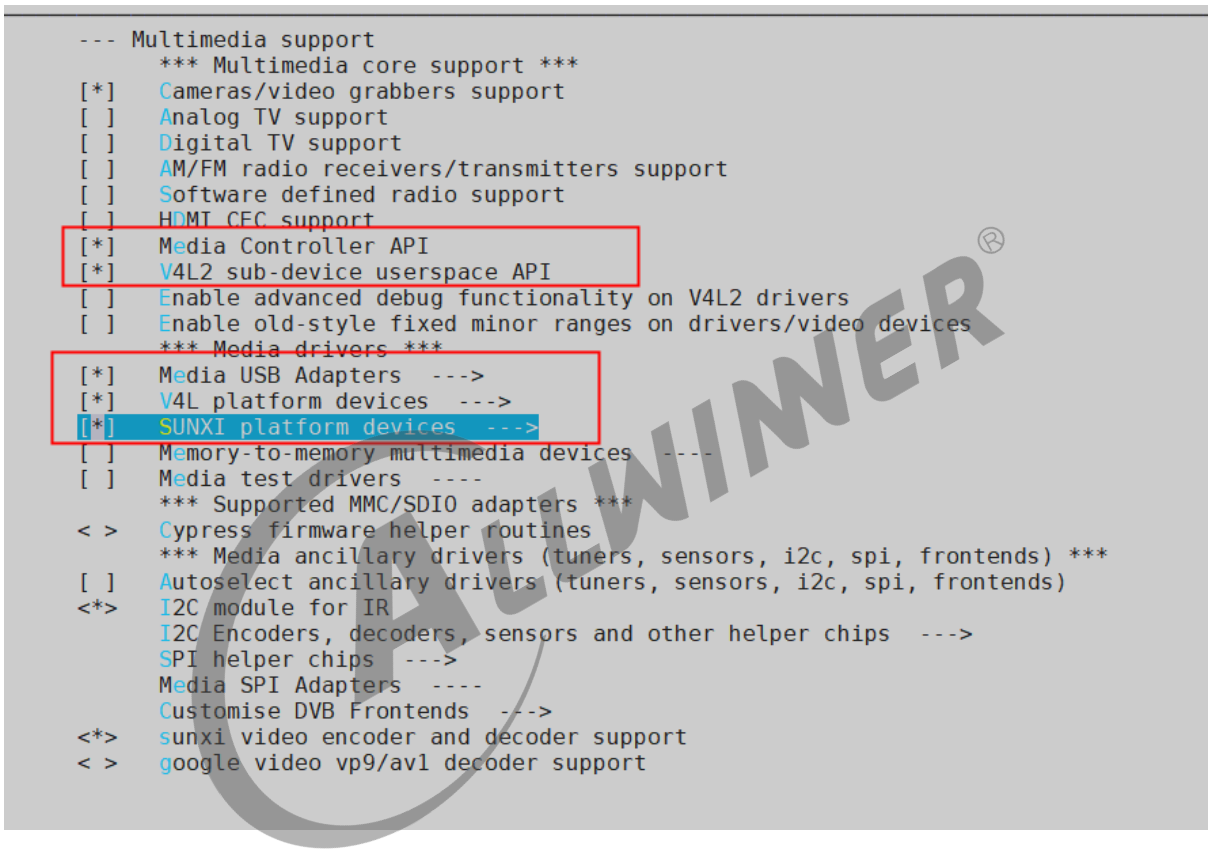

图 2-4: menuconfig_1

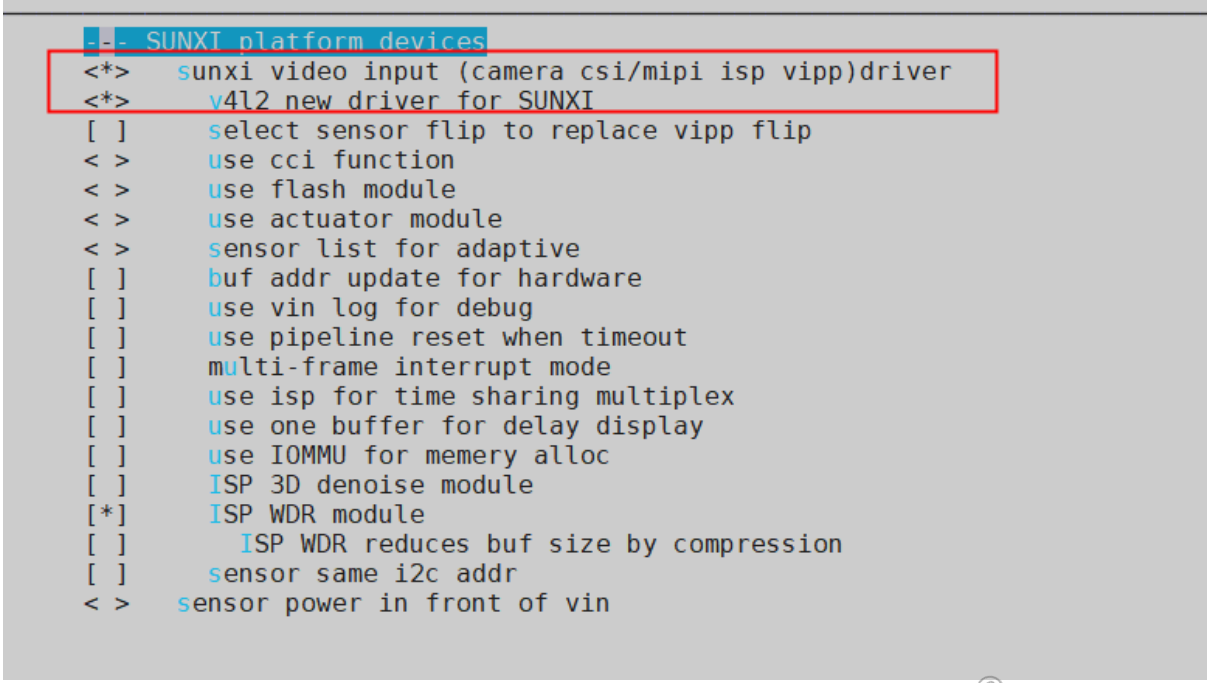

图 2-5: menuconfig_2

2.15.3 软件修改设备树和驱动

需要注意的是，由于 board.dts 中修改较多，V0.8 SDK 中提交为 board.dts_n5, 直接 pull 下来作为 board.dts 便可。如果需要移植，需要注意复用引脚的使用，这里主要留意的是 n5 使用到的引脚（PE0~PE11,PE13,PB6,PB7）；V1.0 以后的 SDK 则只使用一个 board.dts。

2.15.4 软件修改 Makefile

主要是添加 n5 驱动的编译，修改如下：

图 2-6: Makefile

2.15.5 应用测试

这里使用 csitest 来做测试，具体测试流程如下：

```
# ./csitestT5_ptr 0 0 1280 720 / 4 100 25
INFO    : cedarc <VeInitialize:1307>: *** ic_version = 0x1301000010210,
open /dev/video0 fd = 5
[  302.536265] find n5 csi camera sensor now.
fmt.fmt.pix_mp.pixelformat=825382478
resolution got from sensor = 1280*720 num_planes = 1
VIDIOC_QUERYBUF buf.m.planes[j].length=1382400
VIDIOC_QUERYBUF buf.[  302.554541] [DISP] disp_get_layer,line:111:
m.planes[j].length=1382400
VIDI[  302.554547] disp_get_layer (0,2,0) fail
OC_QUERYBUF buf.m.planes[j].leng[  302.566924] [n5]sensor_s_stream on = 1, 1280*720 2006
th=1382400
VIDIOC_QUERYBUF buf.m.planes[j].length=1382400
VIDIOC_QUERYBUF buf.m.planes[j].length=1382400
VIDIOC_QUERYBUF buf.m.planes[j].length=1382400
VIDIOC_QUERYBUF buf.m.planes[j].length=1382400
VIDIOC_QUERYBUF buf.m.planes[j].length=1382400
[  302.604161] n5_set_chn_720p_25 chn=0
[  302.614696] n5_set_chn_720p_25 chn=1
VIDIOC_STREAMON ok
[  306.658034] [DISP] disp_get_layer,line:111:
[  306.658041] disp_get_layer (0,2,0) fail
[  306.687531] [n5]sensor_s_stream on = 0, 1280*720 2006
VIDIOC_STREAMOFF ok
mode 4 test done at the 0 time!!
time cost 4.206329(s)
```

图 2-7: 执行流程

执行完成后，会在 / 目录下生成对应的 yuv 文件，可以通过 rawviewer 打开。

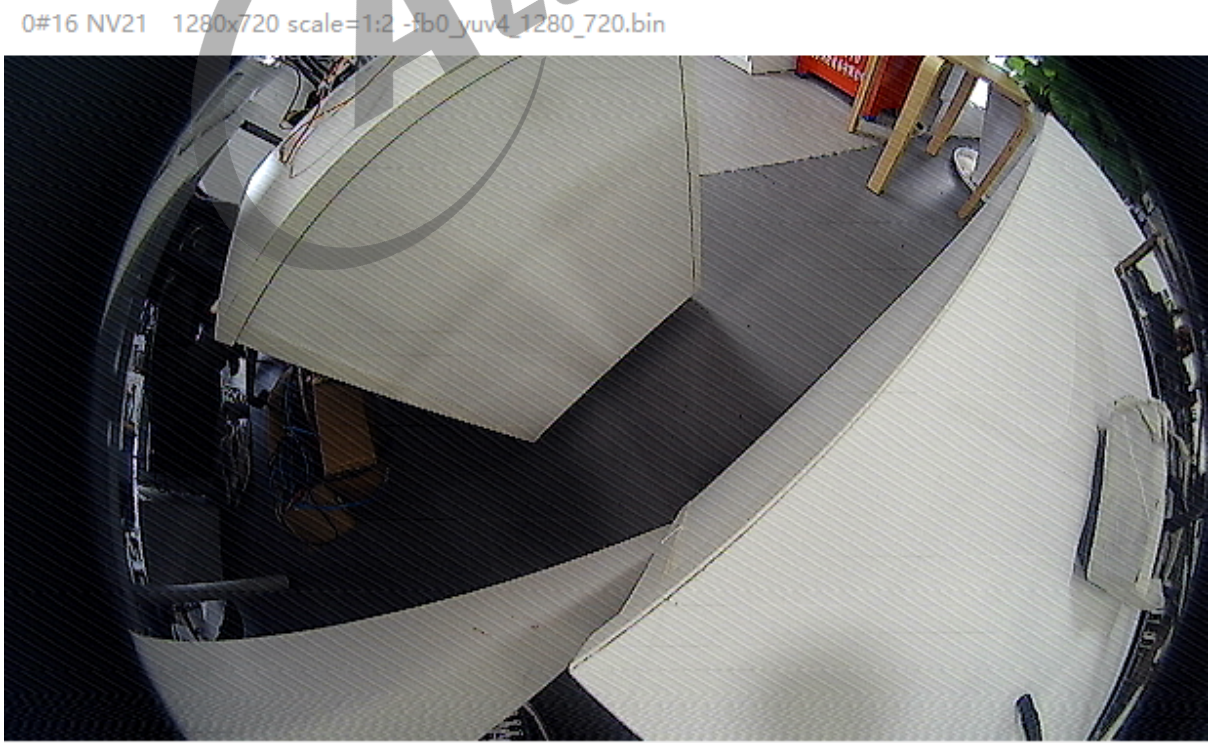

图 2-8: 效果

2.16 过温保护

2.16.1 配置方法

内核配置：

```
arch/arm/configs/sun8iw20p1smp_t113_auto_defconfig
CONFIG_THERMAL=y
CONFIG_THERMAL_EMERGENCY_POWEROFF_DELAY_MS=0
CONFIG_THERMAL_HWMON=y
CONFIG_THERMAL_OF=y
CONFIG_THERMAL_DEFAULT_GOV_STEP_WISE=y
CONFIG_THERMAL_GOV_STEP_WISE=y
CONFIG_THERMAL_GOV_USER_SPACE=y
CONFIG_CPU_THERMAL=y
CONFIG_THERMAL_EMULATION=y
CONFIG_SUNXI_THERMAL=y
CONFIG_WATCHDOG=y
CONFIG_WATCHDOG_CORE=y
```

dts 配置关机温度：

```
T113-S3：关机温度为110摄氏度
kernel/linux-5.4/arch/arm/boot/dts/sun8iw20p1.dtsi
    thermal-zones {
        cpu_thermal_zone: cpu_thermal_zone {
            polling-delay-passive = <500>;
            polling-delay = <1000>;
            thermal-sensors = <&ths 0>;
            sustainable-power = <589>;

                cpu_crit: cpu_crit@0 {
                    temperature = <110000>;
                    type = "critical";
                    hysteresis = <0>;
                };
            };
        };
    };

T113-i：关机温度为115摄氏度
device/config/chips/t113_i/configs/evb1_auto/linux-5.4/board.dts
    &cpu_thermal_zone {
        cpu_trips: trips {
            cpu_crit: cpu_crit@0 {
                temperature = <115000>;
                type = "critical";
                hysteresis = <0>;
            };
        };
    };
```

2.16.2 测试方法（以 T113-i 为例）

```
查看当前温度：
# cat /sys/class/thermal/thermal_zone0/temp
39061
```

模拟当前温度为 114℃，结果：不会关机。

```
echo 114000 > /sys/class/thermal/thermal_zone0/emul_temp
```

模拟当前温度为 116℃，结果：直接关机，只有拔电源才会恢复。

```
echo 116000 > /sys/class/thermal/thermal_zone0/emul_temp
```

2.17 SPI-NAND 配置

要使用 SPI NAND，在配置 SDK 时选择 SPI NAND 板型并按下文修改硬件即可，配置 SDK 具体参考本文档编译各种板型如何配置 SDK章节内容，如果需要自行配置新板型，参考如下内容。

flash 如果选用 SPI NAND 的话，需要在 board.dts 把 sdc2 禁止 (因为引脚有复用），然后打开 spi0。如下：

```
&sdc2 {
    non-removable;
    bus-width = <4>;
    mmc-ddr-1_8v;
    mmc-hs200-1_8v;
    no-sdio;
    no-sd;
    ctl-spec-caps = <0x308>;
    cap-mmc-highspeed;
    sunxi-power-save-mode;
    sunxi-dis-signal-vol-sw;
    mmc-bootpart-noacc;
    max-frequency = <150000000>;
    /*vmmc-supply = <&reg_dcdc1>;*/
    /*emmc io vol 3.3v*/
    /*vqmmc-supply = <&reg_aldo1>;*/
    /*emmc io vol 1.8v*/
    /*vqmmc-supply = <&reg_eldo1>;*/
    status = "disabled";
};
```

```
&spi0 {
    clock-frequency = <100000000>;
    pinctrl-0 = <&spi0_pins_a &spi0_pins_b>;
    pinctrl-1 = <&spi0_pins_c>;
    pinctrl-names = "default", "sleep";
    /*spi-supply = <&reg_dcdc1>;*/
    spi_slave_mode = <0>;
```

```
    spi0_cs_number = <1>;
        spi0_cs_bitmap = <1>;
    status = "okay";

    spi-nand@0 {
        compatible = "spi-nand";
        spi-max-frequency=<0x5F5E100>;
        reg = <0x0>;
        spi-rx-bus-width=<0x04>;
        spi-tx-bus-width=<0x04>;
        status="okay";
    };
};
```

然后再把 sys_config.fex 中的：

```
[target]
storage_type    = 5
```

若出现如下提示，则应该是贴了一个没有适配过的新物料：

```
try emmc fail
[02.718]sunxi-spinand: AW SPINand MTD Layer Version: 1.5 20200407
[02.724]sunxi-spinand-phy: AW SPINand Phy Layer Version: 1.9 20200306
[02.731]sunxi-spinand-phy: request spi0 gpio ok
[02.735]sunxi-spinand-phy: request general tx dma channel ok!
[02.741]sunxi-spinand-phy: request general rx dma channel ok!
[02.746]sunxi-spinand-phy: set spic0 clk to 20 Mhz
[02.751]sunxi-spinand-phy: init spic0 clk ok
[02.755]sunxi-spinand-phy: not detect any munufacture from id table
[02.761]sunxi-spinand-phy: get spi-nand Model from fdt fail
[02.766]sunxi-spinand-phy: get phy info from fdt fail
[02.771]sunxi-spinand-phy: not detect munufacture from fdt
[02.776]sunxi-spinand-phy: detect munufacture from id table: XTX
[02.782]sunxi-spinand-phy: get spi-nand Model from fdt fail
[02.787]sunxi-spinand-phy: get phy info from fdt fail
[02.792]sunxi-spinand-phy: not detect munufacture from fdt
[02.797]sunxi-spinand-phy: not match spinand: 1100110b 11001100
try nand fail
```

需要对着规格书配置 id 表，比如某新增物料的 patch 如下：

```
brandy-2.0/u-boot-2018 / drivers/mtd/awnand/spinand/physic/id.c
    {
        /* XT26G01C */
        .Model      = "XT26G01C",
        .NandID     = {0x0B, 0x11, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
        .DieCntPerChip  = 1,
        .SectCntPerPage = 4,
        .PageCntPerBlk  = 64,
        .BlkCntPerDie   = 1024,
        .OobSizePerPage = 64,
        .OperationOpt   = SPINAND_QUAD_READ | SPINAND_QUAD_PROGRAM |
            SPINAND_DUAL_READ,
        .MaxEraseTimes  = 50000,
        .ecc_status_shift = ECC_STATUS_SHIFT_4,
        .EccType    = BIT4_LIMIT5_TO_8_ERR9_TO_15,
```

```
        .EccProtectedType = SIZE16_OFF0_LEN16,
        .BadBlockFlag    = BAD_BLK_FLAG_FRIST_1_PAGE,
    },
```

重编 uboot：

```
./build.sh bootloader
```

如果配成功后，记得去内核同样的目录加上对应的 id 参数，整个系统就都支持上了。

IO 冲突与硬件修改：

1.SPI NAND 与 EMMC、SPI NOR 三种存储的 IO 口复用 PC2-PC7，不能同时使用;

2.T113-S3 evb 板硬件修改：连接 R15、R17、R19、R21、R23、R25；断开 R14、R16、R18、R20、R22、R24；断开 R874、R875、R876、R877、R878、R879。

T113-S3 SPI NAND 相关硬件跳线截图：

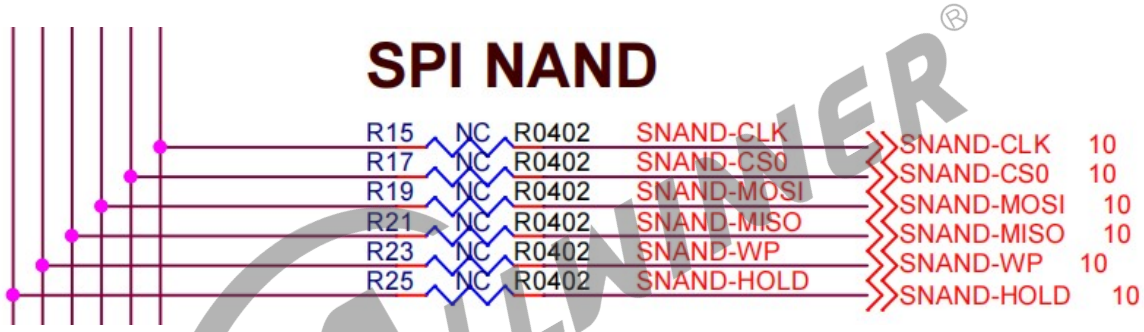

图 2-9: 原理图

3.T113-i evb 板硬件修改：连接 R15、R17、R19、R21、R23、R25；断开 R283、R16、R18、R20、R22、R24；断开 R874、R875、R876、R877、R878、R879。

2.18 SPI NOR 相关问题

2.18.1 调试串口问题

T113-S3 V0.8 版本 SDK 的 nor 板型与 EMMC 板型在同一目录，sys_config.fex 和 board.dts 是共用的，但是 env.cfg、uboot 的 defconfig 等配置文件又不同，所以使用 nor 需要确保调试串口所有地方配置一致，并且与硬件相符，按照前面使能 TF 卡章节修改调试串口的方法进行修改。

T113-S3 V1.0 以后版本 SDK 以及 T113-i SDK 中的 SPI NOR 的配置已经独立出来，位置：device/config/chips/t113/configs/evb1_auto_nor 以及 device/config/chips/t113_i/configs/evb1_auto_nor。

2.18.2 “distro_bootcmd” not defined 问题

uboot 阶段报错如下：

```
** Unrecognized filesystem type **
[00.309]sunxi bmp info error : unable to open logo file bootlogo.bmp
[00.318]Loading Environment from SUNXI_FLASH... *** Warning - bad CRC, using default
    environment

*** Warning - no device, using default environment

Failed (-5)
[00.355]update part info
[00.358]update bootcmd
[00.360]change working_fdt 0x43f0ae70 to 0x43eeae70
disable nand error: FDT_ERR_NOTFOUND
[00.384]update dts
Hit any key to stop autoboot:  2 FDT ERROR:get muxsel err returned FDT_ERR_INTERNAL
sunxi_pwm_pin_set_state, fdt_set_all_pin, ret=-1
[00.487]LCD open finish
                                                        0
## Error: "distro_bootcmd" not defined
=>
```

这个的错误原因是 Boardconfig_nor.mk 配了`LICHEE_REDUNDANT_ENV_SIZE:=0x20000`，导致.buildconfig 里`export LICHEE_REDUNDANT_ENV_SIZE`到了环境变量进一步导致 uboot 编译的时候用了这个，出了问题。

改正方法：

- 排查.mk/.buildconfig 里不要有 LICHEE_REDUNDANT_ENV_SIZE
- 重新开一个编译 shell，确保 env 没有这个变量。

IO 冲突与硬件修改：

1.SPI NOR 与 EMMC、SPI NAND 三种存储的 IO 口复用 PC2-PC7，不能同时使用;

2.T113-S3 evb 板硬件修改：连接 R15、R17、R19、R21、R23、R25；断开 R14、R16、R18、R20、R22、R24；断开 R874、R875、R876、R877、R878、R879。

3.T113-i evb 板硬件修改：连接 R15、R17、R19、R21、R23、R25；断开 R283、R16、R18、R20、R22、R24；断开 R874、R875、R876、R877、R878、R879。

2.19 以太网配置与使用

2.19.1 以太网配置

T113-S3 的 SPI NAND 板型默认已经使能以太网，不需要软件修改配置，T113-i 的 SPI NAND 板型配置中依旧使能的是 Camera 模块，需要修改配置使能以太网。

注意：由于以太网与 Camera 冲突，所以以太网在 dts 中状态默认是 disabled。需要自行修改 dts 开启。

文件路径`device/config/chips/t113/configs/evb1_auto/linux-5.4/board.dts`，将 gmac0 状态改为 okay，修改如下：

```
&gmac0 {
        pinctrl-0 = <&gmac0_pins_a>;
        pinctrl-1 = <&gmac0_pins_b>;
        pinctrl-names = "default", "sleep";
        phy-mode = "rmii";
        use_ephy25m = <1>;
        tx-delay = <7>;
        rx-delay = <31>;
        phy-rst = <&pio PE 13 GPIO_ACTIVE_HIGH>;
        status = "okay";
};
```

为解决 IO 冲突问题，还需要将 Camera 关闭，也就是将 vind0 这个节点的状态改为 disabled：

```
&vind0 {
        csi_top = <378000000>;
        csi_isp = <327000000>;
        status = "disabled";

        actuator0: actuator@5809450 {
```

IO 冲突与硬件修改：

1. 以太网与 Camera 冲突，不能同时使用。

2.T113-S3 目前使用外部 25M 时钟，需要修改硬件连接 RN5 、RN6 、CN4 、CN5 ，并在 YN1 上焊接 25M 晶振。

3.T113-i 可直接播码打开 SW10 与 SW11，关闭 SW8 和 SW9，同时默认使用外部 25M 时钟，不需要修改硬件。

2.19.2 以太网测试

烧录固件后，在开发板输入如下命令进行以太网测试：

```
ifconfig eth0 up
udhcpc -i eth0
ping www.baidu.com
```

2.20 wifi 配置与使用

T113-i evb 和 T113-S3 evb 均默认使用 xr829。

T113-i sdk 已默认配好，boart.dts 配置如下：

```
 rfkill: rfkill@0 {
        compatible    = "allwinner,sunxi-rfkill";
        chip_en;
        power_en;
        pinctrl-0 = <&wlan_pins_a>;
        pinctrl-names = "default";
        status        = "okay";

        wlan: wlan@0 {
            compatible    = "allwinner,sunxi-wlan";
            clock-names = "32k-fanout1";
            clocks = <&ccu CLK_FANOUT1_OUT>;
            wlan_busnum    = <0x1>;
            wlan_regon    = <&pio PD 15 GPIO_ACTIVE_HIGH>;
            wlan_hostwake  = <&pio PG 10 GPIO_ACTIVE_HIGH>;
            /*wlan_power     = "VCC-3V3";*/
            /*wlan_power_vol = <3300000>;*/
            /*interrupt-parent = <&pio>;
            interrupts = < PG 10 IRQ_TYPE_LEVEL_HIGH>;*/
            wakeup-source;

        };
```

IO 冲突与硬件修改：

T113-i evb 板的 SW4 第 4 位需拨到 ON 位置，该位为 PD15：WL-RESETN 脚位，同时使用 wifi 时需要保证 sw2 的第 8 位不被拨到 ON 上 (LCD)，以及 R931 电阻需要为 NC 状态。

2.20.1 wifi 测试

STA 模式：

```
1.手机释放热点或路由器释放热点，记录下ssid

2.终端中输入:/etc/wlan-connect.sh ssid psk mode

3.ifconfig 看到wlan0获取到ip则成功连上AP并dhcp成功

说明:

ssid : WiFi热点名称，只支持2.4G频段
```

```
psk : wifi密码

mode : 0: 无密码，1:WPA/WPA2密码，2:wep密码模式

例: /etc/wlan-connect.sh wally 123456 1
```

AP 模式：

```
1.终端输入 /etc/wlan-softap.sh

2.手机搜索wifi,默认配置为ssid:aw_ap001，psk:12345678

3.连上后手机会有相应提示

注: 如果要修改释放的AP名和密码，需要修改配置文件 /etc/hostapd.conf中ssid和wpa_passphrase
```

2.21 BT 配置与使用

T113-i evb 和 T113-S3 evb 均默认使用 xr829。

T113-i sdk 已默认配好，boart.dts 配置如下：

```
boarts.dts修改

    bt: bt@0 {
        compatible    = "allwinner,sunxi-bt";
        pinctrl-0 = <&wlan_pins_a>;
        pinctrl-names = "default";
        clock-names = "32k-fanout1";
        clocks = <&ccu CLK_FANOUT1_OUT>;
        /*bt_power_num = <0x01>;*/
        /*bt_power      = "axp803-dldo1";*/
        /*bt_io_regulator = "axp803-dldo1";*/
        /*bt_io_vol = <3300000>;*/
        /*bt_power_vol = <330000>;*/
        bt_rst_n      = <&pio PD 14 GPIO_ACTIVE_LOW>;
        status        = "okay";
    };
};

btlpm: btlpm@0 {
    compatible  = "allwinner,sunxi-btlpm";
    uart_index  = <0x1>;
    bt_wake     = <&pio PD 13 GPIO_ACTIVE_HIGH>;
    bt_hostwake = <&pio PD 12 GPIO_ACTIVE_HIGH>;
    status      = "okay";
};
```

IO 冲突与硬件修改：

T113-i evb 开发板 SW4 第 1,2,3 位为 PD12~14，需要拨动到 ON 位置，使用 bt 时需要保证

sw2 的第 5,6,7 位不被拨到 ON 上 (LCD)，以及 R896 R928 R930 电阻为 NC 状态。

2.21.1 BT 测试 (蓝牙收发音乐为例)

sink 模式 (手机作为源通过蓝牙播放音乐，音乐从本机耳机孔出来):

```
1.insmod /lib/modules/5.4.61+/xradio_btlpm.ko
2.echo 0 > /sys/class/rfkill/rfkill0/state
3.echo 1 > /sys/class/rfkill/rfkill0/state
4.hciattach -n ttyAS1 xradio > /dev/null 2>&1 &
5.bluetoothd -n &
6.bluealsa -p a2dp-sink &
7.手机上打开蓝牙
8.bluetoothctl
9.menu advertise
10.name *                                          // 广播的蓝牙名
11.discoverable on
12.back
13.advertise on
14.agent on
15.default-agent
16.scan on
17.trust **:**:**:**                               // 想要配对的蓝牙mac,记录下mac
18.pair **:**:**:**                                // 想要配对的蓝牙mac
19.connect **:**:**:**                             // 想要配对的蓝牙mac
20.exit                                            // 退出bluetoothctl
21.bluealsa-aplay --pcm=hw:audiocodec **:**:**:**  //配对的蓝牙mac
22.已配对上的设备放歌，一般为手机
```

source 模式 (本机作为源，通过蓝牙播放蓝牙音箱):

```
1.insmod /lib/modules/5.4.61+/xradio_btlpm.ko
2.echo 0 > /sys/class/rfkill/rfkill0/state
3.echo 1 > /sys/class/rfkill/rfkill0/state
4.hciattach -n ttyAS1 xradio > /dev/null 2>&1 &
5.bluetoothd -n &
6.bluealsa -p a2dp-source &
7.打开蓝牙音响
8.bluetoothctl
9.menu advertise
10.name *                                      // 广播的蓝牙名
11.discoverable on
12.back
13.advertise on
14.agent on
15.default-agent
16.scan on
17.trust **:**:**:**                            //(想要配对的蓝牙mac,记录下mac,这一步为信任设备)
18.pair **:**:**:**                            //(想要配对的蓝牙mac)
19.connect **:**:**:**                         //(想要配对的蓝牙mac)
20.exit                                        //退出bluetoothctl
21.aplay -D bluealsa:HCI=hci0,DEV=**:**:**:**,PROFILE=a2dp /etc/3.wav
```

2.22 RS485 配置与使用

T113-I RS485 使用串口 1 进行通讯，board.dts 配置如下:

```
&uart1 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&uart1_pins_a>;
    pinctrl-1 = <&uart1_pins_b>;
    status = "okay";
};
```

IO 冲突与硬件修改：T113-i evb 板拨码开关 SW4 第 6 位需要拨向 ON 位置，PD17 为 RS485_EN；

发送端口：RE/DE 强制拉高；接收端：RE/DE 强制拉低；

R819，R820 贴上电阻或短接；R946，R947 拆除；R819&R820：连上 RS485 两个脚位（DO 和 DI）；拆除 R946&R947（T113-I 默认使用 uart1 连接 xr829 进行蓝牙通讯，需要用 RS485 时则需要拆除这两个电阻）。

2.22.1 RS485 测试

T113-i 如要测试 RS485，需要两块板子，或者对端板子 RS485 芯片同样为型号 SP3485EN。

测试方法如下:

```
两块板子485 AB两路对接后(A接A，B接B)
发送端口: echo 0x** > /dev/ttyAS1
接收端:cat /dev/ttyAS1

例:
开发板A: echo 0xAA > /dev/ttyAS1
开发板B: cat /dev/ttyAS1
```

开发板 B 收到 0xAA，表示 RS485 数据链路已通。

2.23 OTA 功能使用

OTA 功能主要是新增一个 rootfsbak 分区，可以使用升级程序将固件升级到 rootfsbak 分区中。使用方法如下。

首先修改`device/config/chips/t113/configs/evb1_auto/longan/sys_partition.fex` ，新增一个分区，大小与 rootfs 相同。新增分区可以放在 rootfs 后面，也可以放在最后，这个会影响后面升级命令所使用的块设备节点名字。

```
diff --git a/configs/evb1_auto/longan/sys_partition.fex b/configs/evb1_auto/longan/
    sys_partition.fex
index 942805e..087d297 100755
--- a/configs/evb1_auto/longan/sys_partition.fex
+++ b/configs/evb1_auto/longan/sys_partition.fex
@@ -65,6 +65,11 @@ size = 16384
     user_type    = 0x8000

 [partition]
+    name         = rootfsbak
+    size         = 2097152
+    user_type    = 0x8000
+
+[partition]
     name         = dsp0
     size         = 2048
     downloadfile = "dsp0.fex"
```

然后重新编译固件，烧录。

当需要 OTA 升级时，首先编译升级工具：

```
cd platform/framework/auto/ota
make
```

将编译生成的可执行程序 upgrade 文件 push 进开发板。

然后按正常流程编译固件，生成的固件同样 push 进开发板，放到可执行程序 upgrade 的同级目录。当前 rootfs 在/dev/mmcblk0p5；rootfsbak 在/dev/mmcblk0p6。cd 进去存放文件的目录，执行如下升级命令：

```
./upgrade d_t113_linux_evb1_auto_uart3_ota.img mmcblk0p6
```

当更新完毕，看到`system OTA successfull!`就表示更新成功，新的固件被更新到了/dev/mmcblk0p6。

在 OTA 成功后，在系统启动时，可以看到:

```
EXT4-fs (mmcblk0p6): mounted filesystem
```

这里出现了 mmcblk0p6 表明 OTA 成功，rootfs 已经变为/dev/mmcblk0p6。

2.24 OTA AB 系统升级

编译及使用方法如下。

编译之前，先删除 out 目录，在 SDK 根目录执行如下命令：

```
source build/envsetup.sh
ota_ab
```

如果提示没有这个命令，说明前面步骤没有成功，然后执行如下命令编译并打包固件：

```
./build.sh
./build.sh pack
```

烧入打包好的固件到机器中，当需要升级时，输入以下命令生成升级固件：

```
swupdate_pack_swu -ab
```

会提示生成 t113_evb1_auto-ab.swu 固件以及路径，把 t113_evb1_auto-ab.swu 固件使用 adb push 到机器的/mnt 目录下，在机器中执行命令：

```
/etc/swupdate_cmd.sh -i /mnt/t113_evb1_auto-ab.swu -e stable,now_A_next_B
```

上述命令表示固件升级到 B 系统中，升级成功后，会自动重启，如果升级 fail，会一直报错。

重启后，输入命令查看当前所在系统：

```
fw_printenv root_partition
fw_printenv boot_partition
```

如果打印 bootB 和 rootfsB，说明已经 OTA 成功，并且处于 B 系统。

当需要由 B 向 A 系统升级时，操作类似。

2.25 CAN 配置与使用

2.25.1 配置方法

前提：准备两台机器，分别标记为 A,B。

在 board.dts 中启用 can0 配置，can0 节点的 status 从 “disabled” 改为 “okay” 即可。can1 同理。

```
+++ b/configs/evb1_auto/linux-5.4/board.dts
@@ -43,7 +43,7 @@
                compatible = "allwinner,sun8i-can";
                device_type = "can0";
                id = <0>;
-               status = "disabled";
+               status = "okay";
        };
```

引脚连接：同名管脚相连，例如 A 和 B 进行通信，A 的 can0H 接 B 的 can0H；A 的 can0L 接 B 的 can0L；can1 同理。

A/B can0 设置波特率：

```
ip -detail link show can0
ip link set can0 down
ip link set can0 type can bitrate 100000
ip -detail link show can0
ip link set can0 up
```

A/B can0 作为接收端：

```
#RX状态
candump can0 &
```

A/B can0 作为发送端：

```
#测试TX
cansend can0 344#01220000
```

如果要测试 can1 ，上述指令中的 “can0” 替换为 “can1” 即可。注意，仅启用 can0 或 can1 的情况下，指令中使用的都是 “can0”。

2.25.2 测试方法

示例：A can0 与 B can 0 相互收发。

```
A can0 接收，B can0 发送
预期结果：A can0 打印
can0  344   [4]  01 22 00 00
```

```
A can0 发送，B can0 接收
预期结果：A can0 打印
can0  344   [4]  01 22 00 00
```

2.26 休眠唤醒

使用 rtc 时钟作为唤醒源进行休眠唤醒测试：

```
#使用rtc 定时器作为唤醒源
echo "+5" >/sys/class/rtc/rtc0/wakealarm;echo mem > /sys/power/state
#两条命令输入时要紧凑，计时是从指令生效开始的，两条指令间的间隔太长就无效了
```

有如下输出则表明功能正常：

```
# 休眠后的打印
[  142.167912] PM: suspend entry (deep)
...
[  142.208161] Freezing remaining freezable tasks ... (elapsed 0.001 seconds) done.

# 5秒钟之后唤醒输出打印
[  142.226371] hci: ERR: hcd is null
```

```
...
[  142.406366] sunxi-mmc 4022000.sdmmc: sdc set ios:clk 0Hz bm PP pm OFF vdd 0 width 1
    timing LEGACY(SDR12) dt B
```

3 常见问题-FAQ

3.1 编译各种板型如何配置 SDK

这个在已经发布的文档《T113_Longan_Linux_SDK 开发环境配置手册.pdf》中有详细指导，下面给出常规配置命令：

```
EMMC板型：
./build.sh autoconfig -c linux -o longan -k linux-5.4  -i t113 -b evb1_auto -n default
SPI NAND板型：
./build.sh autoconfig -c linux -o longan -k linux-5.4 -i t113 -b evb1_auto_nand -n default
SPI NOR板型：
T113-S3 V0.8版本SDK：  ./build.sh autoconfig -c linux -o bsp -k linux-5.4  -i t113 -b
    evb1_auto -n nor
T113 V1.0以后版本SDK：./build.sh autoconfig -c linux -o bsp -k linux-5.4  -i t113 -b
    evb1_auto_nor -n nor
```

3.2 各板型默认支持哪些功能

EMMC 板型默认支持的功能有：

- EMMC 启动, LVDS LCD, 录像, CSI camera, USB camera, U 盘, SD Card, RTP, AudioCodec

EMMC 板型，以下功能需要修改 board.dts 来开启，并需要关闭冲突模块：

- MIPI 屏, 以太网, RGB 屏, WiFi, BT, I2S1, I2S2, CTP

EMMC 板型，以下功能为纯软件功能，默认都支持：

- recovery, AB 系统, OTA

SPI NAND 板型，默认支持的功能有：

- SPI NAND 启动, SD Card, 以太网

SPI NOR 板型，默认支持的功能有：

- SPI NOR 启动

以上各功能模块，需要修改硬件的，已在本文档对应章节注明。

3.3 串口无输出，或打印 log 缺少某一阶段的内容

如果完全无输出，检查调试串口接的地方是否正确。

T113-S3 的 EMMC、SPI NAND 板型，调试串口默认在 uart3（PG8，PG9），在开发板上是 CON3，需要修改硬件连接 R844, R847。SPI NOR 板型，调试串口默认在 uart0。

如果有部分输出，或者输入命令无法执行，说明配置文件有问题，请参考本文档调试串口配置内容核对各配置文件。

著作权声明

商标声明

免责声明