PurPle-Pi-R1接口调试教程手册

PurPle-Pi-R1

接口调试教程手册

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	何伟聪		2022/08/02

接口调试

首先我们先确定我们sdk内核的版本为4.9.84

Plain Text 复制代码

```
# cat kernel/Makefile | more
```

```
VERSION = 4
PATCHLEVEL = 9
SUBLEVEL = 84
EXTRAVERSION =
NAME = Roaring Lionus
```

确认DTS文件

我们进入我们脚本文件Release_to_customer.sh，进行了解kernel、uboot、rootfs。

kernel

Plain Text 复制代码

```
#build kernel
cd ${RELEASEDIR}/kernel
declare -x ARCH="arm"
declare -x CROSS_COMPILE="arm-linux-gnueabihf-"
if [ "${flashtype}" = "nor" ]; then
        if [ "${fastboot}" = "fastboot" ]; then
                make infinity2m_ssc011a_s01a_fastboot_defconfig
        else
                make infinity2m_ssc011a_s01a_minigui_defconfig
        fi
else
        if [ "${fastboot}" = "fastboot" ]; then
                make infinity2m_spinand_ssc011a_s01a_minigui_fastboot_doublenet_defconfig
        else
                make infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig
        fi

fi
```

如果我们的使用的flash是nand的话，我们使用的defconfig是：

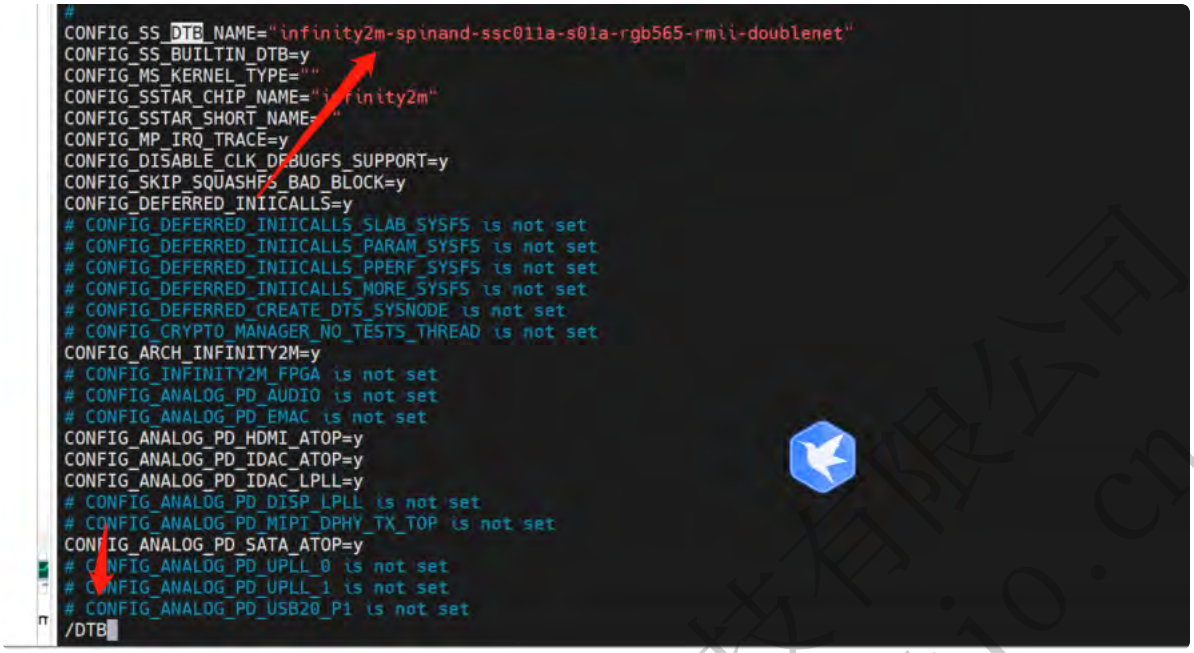
Plain Text 复制代码

```
infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig
```

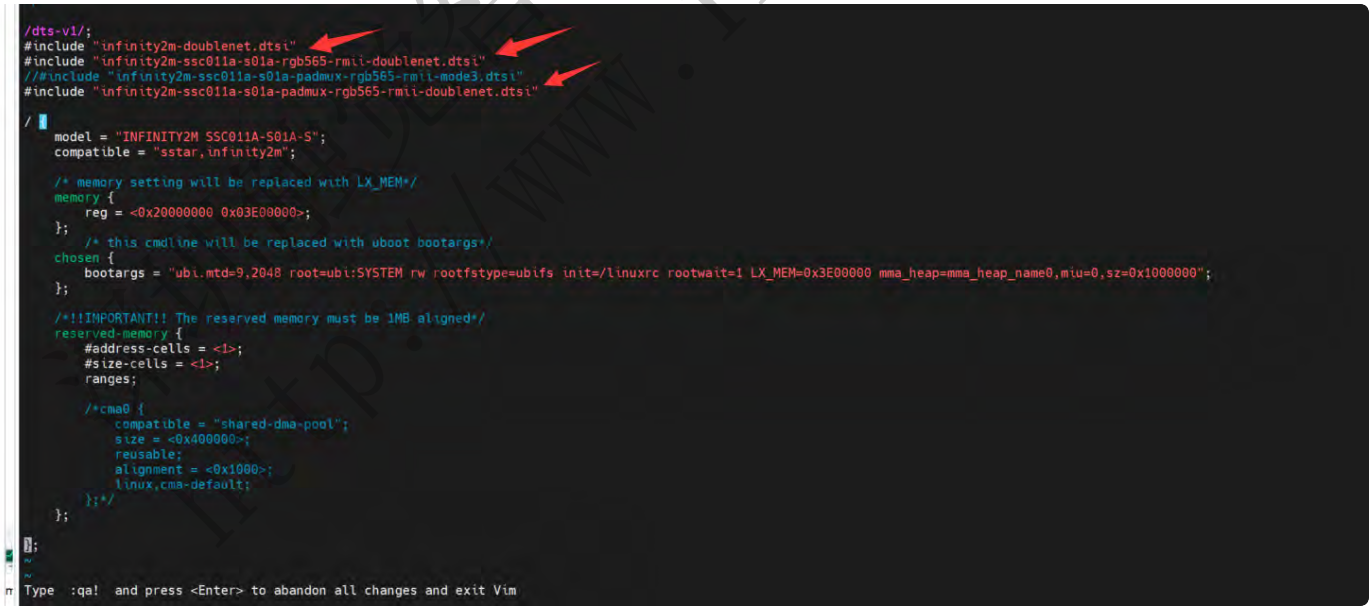
位置在kernel/arch/arm/configs/，我们进入该文件找到生成的DTB文件是：

Plain Text 复制代码

```
infinity2m-spinand-ssc011a-s01a-rgb565-rmii-doublenet
```



从infinity2m-spinand-ssc011a-s01a-rgb565-rmii-doublenet.dtb中可以知道我们的dts就是从infinity2m-spinand-ssc011a-s01a-rgb565-rmii-doublenet.dts，位置在kernel/arch/arm/boot/dts，我们打开dts发现里面包含了这三个文件，我们后续的内核设备树的配置主要都在这三个文件内。



uboot

```
# build uboot
cd ${RELEASEDIR}/boot
declare -x ARCH="arm"
declare -x CROSS_COMPILE="arm-linux-gnueabihf-"
if [ "${flashtype}" = "nor" ]; then
        make infinity2m_defconfig
else
        make infinity2m_spinand_defconfig
fi
#make clean
make -j8

if [ "${flashtype}" = "nor" ]; then
        if [ -d ../project/board/i2m/boot/nor/uboot ]; then
                cp u-boot.xz.img.bin ../project/board/i2m/boot/nor/uboot
        fi
else
        if [ -d ../project/board/i2m/boot/spinand/uboot ]; then
                cp u-boot_spinand.xz.img.bin ../project/board/i2m/boot/spinand/uboot
        fi
fi

```

在Release_to_customer.sh中可以看到，当flash为nand时，uboot下的配置文件如下：位置在/boot/configs/infinity2m_spinand_defconfig

```
vi CONFIG_MS_SAVE_ENV_IN_NAND_FLASH
```

infinity2m_spinand_defconfig配置如下所示：

```
CONFIG_SYS_ARCH="arm"
CONFIG_SYS_CPU="armv7"
CONFIG_SYS_SOC="infinity2m"
CONFIG_SYS_CONFIG_NAME="infinity2m"

```

从CONFIG_SYS_SOC，我们可以知道，我们的配置文件文件为：

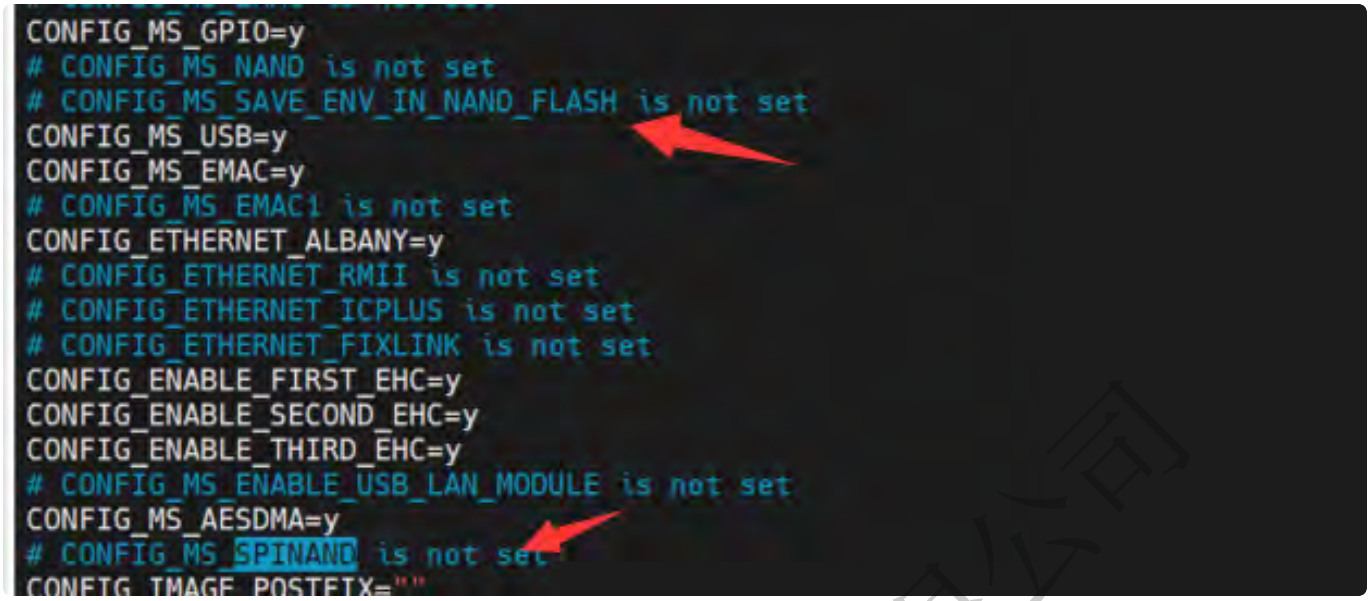
Plain Text 复制代码

```
boot/include/configs/infinity2m.h
```

配置信息如下：

Plain Text 复制代码

```
#ifdef CONFIG_MS_SPINAND
#if defined(CONFIG_MS_SAVE_ENV_IN_NAND_FLASH)
#define CONFIG_ENV_IS_IN_NAND
#define CONFIG_ENV_OFFSET       CONFIG_MSTAR_ENV_NAND_OFFSET
#define CONFIG_MSTAR_ENV_NAND_OFFSET ms_nand_env_offset
/*#define CONFIG_MSTAR_ENV_NAND_OFFSET 0x440000*/
#define CONFIG_ENV_RANGE        0x20000
#define CONFIG_ENV_SIZE         0x1000 // Using 4K length for env is enough, this length must be the same as IPL's env when using fastboot. // 0x00020000
#define CONFIG_ENV_OFFSET_REDUND CONFIG_MSTAR_ENV_NAND_REDUND_OFFSET
#define CONFIG_MSTAR_ENV_NAND_REDUND_OFFSET ms_nand_env_redund_offset
#endif

#define CONFIG_CMD_SPINAND_CIS
#define CONFIG_CMD_UBI
/* #define CONFIG_CMD_UBIFS */
#define CONFIG_UBI_MWRITE
#define MTDIDS_DEFAULT                  "nand0=nand0"    /* "nor0=physmap-flash.0,nand0=nand" */
/*      must be different from real partition to test NAND partition function */
#define MTDPARTS_DEFAULT                "mtdparts=nand0:0xC0000@0x140000(NPT),-(UBI)"
/*      #define MTDPARTS_DEFAULT    "mtdparts=nand0:0x60000@0x140000(IPL0),0x60000(IPL1),0x60000(IPL_CUST0),0x60000(IPL_CUST1),0xC0000(UBOOT0),0xC0000(UBOOT1),0x60000(ENV),0x340000(KERNEL),0x340000(RECOVERY),-(UBI)"*/

```



rootfs

从Release_to_customer.sh中可以看到，当flash类型选择为nand，芯片类型选择为ssd201，并且不开启fastboot模式时，在project目录下执行了：

Plain Text 复制代码

```
1  ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.glibc.011a.128
```

Release_to_customer.sh脚本内容如下：

Plain Text 复制代码

```
#build project
cd ${RELEASEDIR}/project
if [ "${flashtype}" = "nor" ]; then
        if [ "${fastboot}" = "fastboot" ]; then
                echo test fastboot
                ./setup_config.sh ./configs/nvr/i2m/8.2.1/nor.glibc-ramfs.011a.64
        else
                if [ "${chip}" = "ssd201" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/nor.glibc-squashfs.011a.64
                fi
                if [ "${chip}" = "ssd202" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/nor.glibc-squashfs.011a.128
                fi
        fi
else
        if [ "${fastboot}" = "fastboot" ]; then
                if [ "${chip}" = "ssd201" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.ram-glibc-squashfs.011a.64
                elif [ "${chip}" = "ssd202" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.ram-glibc-squashfs.011a.128
                fi
        else
                if [ "${chip}" = "ssd201" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.glibc.011a.64
                fi
                if [ "${chip}" = "ssd202" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.glibc.011a.128
                fi
        fi

fi

```

我们进入配置文件：

Plain Text 复制代码

```
vi project/configs/nvr/i2m/8.2.1/spinand.glibc.011a.128
```

1. IMAGE_CONFIG(分区配置)=spinand.ubifs.p2.partition.config
2. CUSTOMER_TAILOR(APP配置)=nvr_i2m_display_glibc_tailor.mk
3. BOOTLOGO_FILE(logo文件名)=sigmastar1024_600.jpg
4. DISP_OUT_NAME(屏幕型号)=SAT070CP50

1.1. IMAGE_CONFIG

Plain Text 复制代码

```
vi project/image/configs/i2m/spinand.ubifs.p2.partition.config
```

Plain Text 复制代码

```
IMAGE_LIST = cis ipl ipl_cust uboot logo kernel rootfs miservice customer appconfigs
OTA_IMAGE_LIST = ipl ipl_cust uboot logo kernel miservice customer appconfigs
FLASH_TYPE = spinand
UBI_MLC_TYPE = 0
PAT_TABLE  = ubi
PHY_TEST = no
#overwrite CIS(BL0,BL1,UBOOT) PBAs
CIS_PBAs = 10 0 0
CIS_COPIES = 5
USR_MOUNT_BLOCKS:=miservice customer appconfigs
ENV_CFG = /dev/mtd6 0x00000 0x1000 0x20000 2
ENV_CFG1 = /dev/mtd7 0x00000 0x1000 0x20000 2

cis$(RESOUCE) = $(IMAGEDIR)/cis.bin
cis$(DATASIZE) = 0x40000
cis$(PGSIZE) = 2k
cis$(COPIES) = $(CIS_COPIES)
cis$(PATSIZE) = 0x140000
cis$(BOOTTAB) = $(ipl$(MTDPART)),$(ipl_cust$(MTDPART)),$(uboot$(MTDPART))
cis$(SYSTAB) = $(key_cust$(MTDPART)),$(logo$(MTDPART)),$(kernel$(MTDPART)),-(UBI)

ipl$(RESOUCE) = $(PROJ_ROOT)/board/$(CHIP)/boot/ipl/IPL.bin
ipl$(DATASIZE) = 0x20000
ipl$(COPIES) = 3
ipl$(BKCOUNT) = 2
ipl$(PATSIZE) = $(call multiplyhex, $(ipl$(COPIES)), $(ipl$(DATASIZE)))
ipl$(PATCOUNT) = 2
ipl$(MTDPART) = $(ipl$(DATASIZE))@$(cis$(PATSIZE))(IPL0)$(ipl$(BKCOUNT)),$(ipl$(DATASIZE))(IPL1)$(ipl$(BKCOUNT))
ipl$(OTABLK) = /dev/mtd0 /dev/mtd1

ipl_cust$(RESOUCE) = $(PROJ_ROOT)/board/$(CHIP)/boot/ipl/IPL_CUST.bin
ipl_cust$(DATASIZE) = 0x20000
ipl_cust$(COPIES) = 3
ipl_cust$(BKCOUNT) = 2
ipl_cust$(PATSIZE) = $(call multiplyhex, $(ipl_cust$(COPIES)), $(ipl_cust$(DATASIZE)))
ipl_cust$(PATCOUNT) = 2
ipl_cust$(MTDPART) = $(ipl_cust$(DATASIZE))(IPL_CUST0)$(ipl_cust$(BKCOUNT)),$(ipl_cust$(DATASIZE))(IPL_CUST1)$(ipl_cust$(BKCOUNT))
ipl_cust$(OTABLK) = /dev/mtd2 /dev/mtd3

```

```
uboot$(RESOUCE) = $(PROJ_ROOT)/board/$(CHIP)/boot/$(FLASH_TYPE)/uboot/u-boot_$(FLASH_TYPE).xz.img.bin
uboot$(DATASIZE) = 0x40000
uboot$(COPIES) = 3
uboot$(BKCOUNT) = 4
uboot$(PATSIZE) = $(call multiplyhex, $(uboot$(COPIES)), $(uboot$(DATASIZE)))
uboot$(PATCOUNT) = 2
uboot$(MTDPART) =$(uboot$(DATASIZE))(UBOOT0)$(uboot$(BKCOUNT)),$(uboot$(DATASIZE))(UBOOT1)$(uboot$(BKCOUNT)),0x20000(ENV0)1,0x20000(ENV1)1
uboot$(OTABLK) = /dev/mtd4 /dev/mtd5

wifi24mclkcmd = mw 1f001cc0 11
wifirstoffcmd = gpio out 71 0
wifirstoncmd = gpio out 71 1

key_cust$(PATSIZE) = 0x20000
key_cust$(MTDPART) = $(key_cust$(PATSIZE))(KEY_CUST)

logo$(RESOUCE) = $(IMAGEDIR)/logo
logo$(PATSIZE) = 0x60000
logo$(MTDPART) = $(logo$(PATSIZE))(LOGO)
logo$(OTABLK) = /dev/mtd9

kernel$(RESOUCE)   = $(PROJ_ROOT)/release/$(PRODUCT)/$(CHIP)/$(BOARD)/$(TOOLCHAIN)/$(TOOLCHAIN_VERSION)/bin/kernel/$(FLASH_TYPE)/uImage.xz
kernel$(PATSIZE)   = 0x500000
kernel$(BOOTENV)   = $(KERNEL_BOOT_ENV)
kernel$(MTDPART)   = $(kernel$(PATSIZE))(KERNEL),$(kernel$(PATSIZE))(RECOVERY)
kernel$(OTABLK) = /dev/mtd10

rootfs$(RESOUCE)   = $(OUTPUTDIR)/rootfs
rootfs$(FSTYPE)    = ubifs
rootfs$(PATSIZE)   = 0x6200000
rootfs$(BOOTENV)  = console=ttyS0,115200 ubi.mtd=UBI,2048 root=ubi:rootfs rw rootfstype=ubifs init=/linuxrc rootwait=1

miservice$(RESOUCE)   = $(OUTPUTDIR)/miservice/config
miservice$(FSTYPE)    = ubifs
miservice$(PATSIZE)   = 0xA00000
miservice$(MOUNTTG)  = /config
miservice$(MOUNTPT)  = ubi0:miservice
miservice$(OPTIONS)   = rw
miservice$(OTABLK)    = /dev/ubi0_1

customer$(RESOUCE)   = $(OUTPUTDIR)/customer
customer$(FSTYPE)    = ubifs
```

```
customer$(PATSIZE)   = 0x6500000
customer$(MOUNTTG)  = /customer
customer$(MOUNTPT)  = ubi0:customer
customer$(OPTIONS)   = rw
customer$(OTABLK)    = /dev/ubi0_2

appconfigs$(RESOUCE)   = $(OUTPUTDIR)/appconfigs
appconfigs$(FSTYPE)    = ubifs
appconfigs$(PATSIZE)   = 0x400000
appconfigs$(MOUNTTG)  = /appconfigs
appconfigs$(MOUNTPT)  = ubi0:appconfigs
appconfigs$(OPTIONS)   = rw
appconfigs$(OTABLK)    = /dev/ubi0_3
```

CIS： SPI-NAND 独有的分区，保存在 flash 0 地址的位置，它包含两部分内容，一部分是 spinand info，保存 spinand 的一 些基本信息，例如 block page count、block count 等，这些信息会根据 spi nand 的种类而改变，另一部分是 partinifo，保 存的分区信息，这些信息都是给 rom code 读取的，rom code 通过读取正确的 spinand 参数和分区信息，从而知道 IPL 的位置把整个系统跑起来。

IPL: IPL 分区的作用与 SPI-NOR 分区一样，但是从上图可知，IPL 在 spi-nand 上保存了六份，每个 block 一份，目的是为

了防止坏块而做的备份。

IPL_CUS： 作用同 SPI-NOR，会有两个分区，IPL_CUS0, IPL_CUS1,每个分区中各三份 IPL_CUS 的 data。

UBOOT： UBOOT 的二进制文件存放分区, 会有两个分区，每个分区中一份 data。

ENV： UBOOT 的环境变量存放分区。

KERNEL：存放内核的二进制文件。

LOGO: 在 NVR 设备上会使用，存放的是开机 logo 相关的配置。

ROOTFS： rootfs配置

UBI: UBI的内容在上图分区表中不会显示出来，UBI中会创建多个ubifs格式的子分区，客户可以根据需要自行创建。Spinand 的 miservice 分区就是放在 UBI 中。

以上是 BOOT 相关的分区信息，这部分无法任意修改。

1.2. CUSTOMER_TAILOR

Plain Text 复制代码

```
vi project/release/customer_tailor/nvr_i2m_display_glibc_tailor.mk
```

Plain Text 复制代码

```
include $(PROJ_ROOT)/release/customer_tailor/nvr_default.mk
#interface_ai:=disable
#interface_ao:=disable
#interface_gfx:=disable
interface_hdmi:=disable
#interface_divp:=disable
#interface_disp:=disable
#interface_panel:=disable
interface_rgn:=disable
interface_shadow:=disable
interface_uac:=disable
interface_vdf:=disable
interface_vdisp:=disable
interface_vdec:=enable
interface_venc:=enable
interface_wlan:=enable

misc_fbdev:=enable
#verify_zk_full_security:=enable
#mhal
#mhal_aio:=disable
mhal_csi:=disable
#mhal_disp:=disable
#mhal_divp:=disable
mhal_isp:=disable
mhal_ispalgo:=disable
mhal_ispmid:=disable
mhal_ldc:=disable
mhal_mload:=disable
#mhal_panel:=disable
#mhal_rgn:=disable
mhal_sensorif:=disable
#mhal_venc:=disable
mhal_vif:=disable
mhal_vpe:=disable
mhal_hdmitx:=disable

verify_jpeg2disp=enable
verify_disp_init=disable

interface_alsa=enable

```

这个文件是关于一些内核模块的开关和APP的配置。比如interface_wlan，在构建rootfs时会根据它是否enable来加入wifi功能：

Plain Text 复制代码

```
vi project/image/configs/i2m/rootfs.mk
```

```
if [ $(interface_wlan) = "enable" ]; then \
        mkdir -p  $(miservice$(RESOUCE))/wifi ; \
        if [ $(FLASH_TYPE) = "spinand" ]; then \
                cp -rf $(LIB_DIR_PATH)/wifi/libs/ap/*   $(miservice$(RESOUCE))/wifi ; \
                cp -rf $(LIB_DIR_PATH)/wifi/bin/ap/*   $(miservice$(RESOUCE))/wifi ; \
        fi;     \
        find $(LIB_DIR_PATH)/wifi/bin/ -maxdepth 1 -type f -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        find $(LIB_DIR_PATH)/wifi/bin/ -maxdepth 1 -type l -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        find $(LIB_DIR_PATH)/wifi/libs/ -maxdepth 1 -type f -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        find $(LIB_DIR_PATH)/wifi/libs/ -maxdepth 1 -type l -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        cp -rf $(LIB_DIR_PATH)/wifi/modules/*   $(miservice$(RESOUCE))/wifi ; \
        cp -rf $(LIB_DIR_PATH)/wifi/configs/*   $(miservice$(RESOUCE))/wifi ; \
fi;
if [ "$(appconfigs$(RESOUCE))" != "" ]; then \
        if [ -f "$(miservice$(RESOUCE))/wifi/wpa_supplicant.conf" ]; then       \
                mv  $(miservice$(RESOUCE))/wifi/wpa_supplicant.conf $(appconfigs$(RESOUCE));    \
                cp $(OUTPUTDIR)/appconfigs/wpa_supplicant.conf $(appconfigs$(RESOUCE))/wpa_supplicant.conf_bak; \
        fi;     \
        cp -rf $(PROJ_ROOT)/board/$(CHIP)/$(BOARD_NAME)/config/model/LCM.ini $(appconfigs$(RESOUCE));   \
fi;
```

Plain Text 复制代码

```
boot/include/configs/infinity2m.h
```

我们

内存大小问题

Flash分区大小问题

buildroot文件系统定制

ETH

WIFI

LCD

音频配置

UART

GPIO

PWM

I2C

SPI

RTC的配置

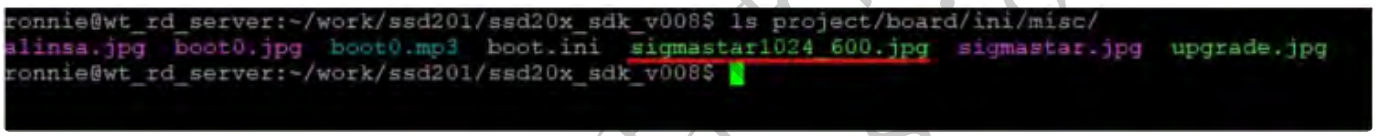
WATCHDOG的配置

QT移植

母片制作

1.3. BOOTLOGO_FILE

可以看到，启动logo图片BOOTLOGO_FILE应该放在project/board/ini/misc/目录下：



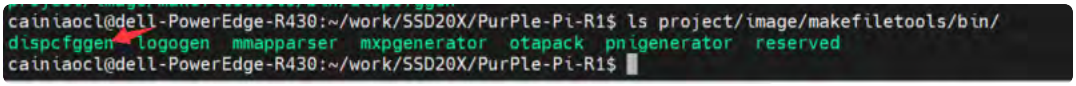
1.4. DISP_OUT_NAME

我们在配置bootlogo的时候，需要指定屏幕的型号，才能有效，而DISP_OUT_NAME是指定的屏幕型号，而disp_data_main.c设置对应的屏幕型号

Plain Text 复制代码

```
1  vi project/image/makefiletools/src/rawgenerator/disp_data_main.c
```

在project/image/makefiletools/bin/dispcfggen用于初始化屏幕



而DISP_OUT_NAME作为参数传递给dispcfggen，而dispcfggen由project/image/makefiletools/src/rawgenerator/disp_data_main.c生成

```
.PHONY: all

TARGET_DISP := dispcfggen
TARGET_LOGO := logogen


all:
	gcc -m32 -Wall ss_raw_header.c logo_data_main.c -o ../../bin/$(TARGET_LOGO)
	gcc -m32 -Wall -I./pnl ss_raw_header.c disp_data_main.c -o ../../bin/$(TARGET_DISP)
```

Plain Text 复制代码

```
1  vi project/configs/nvr/i2m/8.2.1/spinand.glibc.011a.128
```

内存大小的问题

进入Linux系统，使用free –m 命令发现内存小于128M（SSD202的内存大小为128M）：

```
# free -m
              total        used        free      shared  buff/cache   available
Mem:            104          18          79           0           6          83
Swap:             0           0           0
```

了解到内存大小分配给了MMA、Linux系统和一部分reserved（这部分一般不用去修改），即：

DDR total memory = linux memory(cat /proc/meminfo的MemTotal) + mma(mma_heap_name0 + MMU_MMA) + kernel reserved

在Uboot中，可以看到MMA的大小默认设置为0x1000000=16M：

```
bootargs=console=ttyS0,115200 ubi.mtd=UBI,2048 root=ubi:rootfs rw rootfstype=ubifs init=/linuxrc rootwait=1 LX
_MEM=0x7f00000 mma_heap=mma_heap_name0,miu=0,sz=0x1000000 mma_memblock_remove=1 highres=off mmap_reserved=fb,m
iu=0,sz=0x300000,max_start_off=0x7C00000,max_end_off=0x7F00000 mtdparts=nand0:384k@1280k(IPL0),384k(IPL1),384k
(IPL_CUST0),384k(IPL_CUST1),768k(UBOOT0),768k(UBOOT1),256k(ENV),256k(ENV1),0x20000(KEY_CUST),0x60000(LOGO),0x5
00000(KERNEL),0x500000(RECOVERY),-(UBI)
```

因此通过减少MMA的大小来增加Linux系统可用内存，我们先在uboot下修改MMA大小，验证上面的公式：

我们可以直接在：project/configs/nvr/i2m/8.2.1/spinand.glibc.011a.128修改我们的MMA之

```
DISP_OUT_NAME = EQT700BKJ
EXBOOTARGS =
KERNEL_BOOT_ENV = LX_MEM=$(KERNEL_MEMLEN) mma_heap=mma_heap_name0,miu=0,sz=0x1000000 mma_memblock_remove=1 hig
hres=off
TOOLCHAIN_REL = arm-linux-gnueabihf-
```

这里我们所减了6M，对应的linux的内存就增加了6M。

```
# free -m
              total        used        free      shared  buff/cache   available
Mem:            110          22          78           0           9          85
Swap:             0           0           0
```

同时，可以了解到kernel reserved的大小为128M-110-10M=8M。这一部分应该等于0x80000000（128M）- 0x7800000。

地址范围	0x0-0x6E00000	0x6E00000-0x7800000	0x7800000-0x8000000
作用	linux（110M）	mma（10M）	kernel resverved（8M）

Flash分区大小问题

使用df -h 查看分区的情况，我们可以看到rootfs的内存有85.6M：

```
# df -h
Filesystem                Size      Used Available Use% Mounted on
ubi:rootfs               85.6M     32.0M     53.5M  37% /
devtmpfs                 53.6M         0     53.6M   0% /dev
df: /dev/shm: No such file or directory
tmpfs                    53.6M     64.0K     53.5M   0% /tmp
tmpfs                    53.6M     32.0K     53.6M   0% /run
mdev                     53.6M         0     53.6M   0% /dev
ubi0:miservice            7.5M      6.5M   1016.0K  87% /config
ubi0:customer            88.1M    704.0K     87.5M   1% /customer
ubi0:appconfigs           2.0M     24.0K      2.0M   1% /appconfigs
#
```

这个分区是针对SSD202的256M Flash的可以在：project/image/configs/i2m/spinand.ubifs.p2.partition.config_256M查看到分区的情况，注意在脚本里spinand.ubifs.p2.partition.config_256M配置会覆盖spinand.ubifs.p2.partition.config，所以想要修改的话，应该在spinand.ubifs.p2.partition.config_256M修改：

Plain Text 复制代码

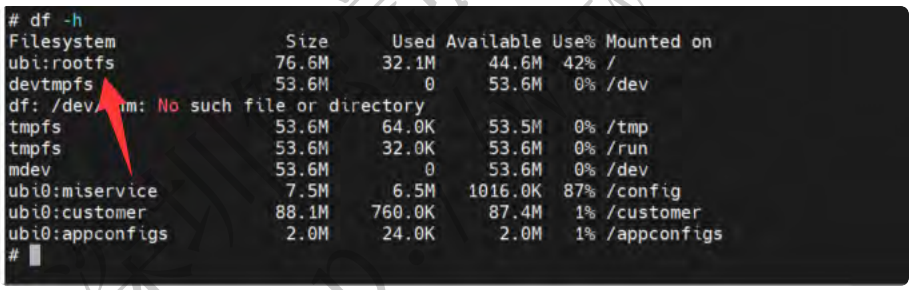
```
cis$(RESOUCE) = $(IMAGEDIR)/cis.bin
cis$(DATASIZE) = 0x40000
cis$(PGSIZE) = 2k
cis$(COPIES) = $(CIS_COPIES)
cis$(PATSIZE) = 0x140000
cis$(BOOTTAB) = $(ipl$(MTDPART)),$(ipl_cust$(MTDPART)),$(uboot$(MTDPART))
cis$(SYSTAB) = $(key_cust$(MTDPART)),$(logo$(MTDPART)),$(kernel$(MTDPART)),-(UBI)

ipl$(RESOUCE) = $(PROJ_ROOT)/board/$(CHIP)/boot/ipl/IPL.bin
ipl$(DATASIZE) = 0x20000
ipl$(COPIES) = 3
ipl$(BKCOUNT) = 2
ipl$(PATSIZE) = $(call multiplyhex, $(ipl$(COPIES)), $(ipl$(DATASIZE)))
ipl$(PATCOUNT) = 2
ipl$(MTDPART) = $(ipl$(DATASIZE))@$(cis$(PATSIZE))(IPL0)$(ipl$(BKCOUNT)),$(ipl$(DATASIZE))(IPL1)$(ipl$(BKCOUNT))
ipl$(OTABLK) = /dev/mtd0 /dev/mtd1

ipl_cust$(RESOUCE) = $(PROJ_ROOT)/board/$(CHIP)/boot/ipl/IPL_CUST.bin
ipl_cust$(DATASIZE) = 0x20000
ipl_cust$(COPIES) = 3
ipl_cust$(BKCOUNT) = 2
ipl_cust$(PATSIZE) = $(call multiplyhex, $(ipl_cust$(COPIES)), $(ipl_cust$(DATASIZE)))
ipl_cust$(PATCOUNT) = 2
ipl_cust$(MTDPART) = $(ipl_cust$(DATASIZE))(IPL_CUST0)$(ipl_cust$(BKCOUNT)),$(ipl_cust$(DATASIZE))(IPL_CUST1)$(ipl_cust$(BKCOUNT))
ipl_cust$(OTABLK) = /dev/mtd2 /dev/mtd3

uboot$(RESOUCE) = $(PROJ_ROOT)/board/$(CHIP)/boot/$(FLASH_TYPE)/uboot/u-boot_$(FLASH_TYPE).xz.img.bin
uboot$(DATASIZE) = 0x40000
uboot$(COPIES) = 3
uboot$(BKCOUNT) = 4
uboot$(PATSIZE) = $(call multiplyhex, $(uboot$(COPIES)), $(uboot$(DATASIZE)))
uboot$(PATCOUNT) = 2
uboot$(MTDPART) =$(uboot$(DATASIZE))(UBOOT0)$(uboot$(BKCOUNT)),$(uboot$(DATASIZE))(UBOOT1)$(uboot$(BKCOUNT)),0x20000(ENV0)1,0x20000(ENV1)1
uboot$(OTABLK) = /dev/mtd4 /dev/mtd5

wifi24mclkcmd = mw 1f001cc0 11
wifirstoffcmd = gpio out 71 0
wifirstoncmd = gpio out 71 1
```

```
key_cust$(PATSIZE) = 0x20000
key_cust$(MTDPART) = $(key_cust$(PATSIZE))(KEY_CUST)

logo$(RESOUCE) = $(IMAGEDIR)/logo
logo$(PATSIZE) = 0x60000
logo$(MTDPART) = $(logo$(PATSIZE))(LOGO)
logo$(OTABLK) = /dev/mtd9

kernel$(RESOUCE)   = $(PROJ_ROOT)/release/$(PRODUCT)/$(CHIP)/$(BOARD)/$(TOOLCHAIN)/$(TOOLCHAIN_VERSION)/bin/kernel/$(FLASH_TYPE)/uImage.xz
kernel$(PATSIZE)   = 0x500000
kernel$(BOOTENV)   = $(KERNEL_BOOT_ENV)
kernel$(MTDPART)   = $(kernel$(PATSIZE))(KERNEL),$(kernel$(PATSIZE))(RECOVERY)
kernel$(OTABLK) = /dev/mtd10

rootfs$(RESOUCE)   = $(OUTPUTDIR)/rootfs
rootfs$(FSTYPE)    = ubifs
rootfs$(PATSIZE)   = 0x6200000
rootfs$(BOOTENV)   = console=ttyS0,115200 ubi.mtd=UBI,2048 root=ubi:rootfs rw rootfstype=ubifs init=/linuxrc rootwait=1

miservice$(RESOUCE)   = $(OUTPUTDIR)/miservice/config
miservice$(FSTYPE)    = ubifs
miservice$(PATSIZE)   = 0xA00000
miservice$(MOUNTTG)  = /config
miservice$(MOUNTPT)  = ubi0:miservice
miservice$(OPTIONS)   = rw
miservice$(OTABLK)    = /dev/ubi0_1

customer$(RESOUCE)   = $(OUTPUTDIR)/customer
customer$(FSTYPE)    = ubifs
customer$(PATSIZE)   = 0x6500000
customer$(MOUNTTG)  = /customer
customer$(MOUNTPT)  = ubi0:customer
customer$(OPTIONS)   = rw
customer$(OTABLK)    = /dev/ubi0_2

appconfigs$(RESOUCE)   = $(OUTPUTDIR)/appconfigs
appconfigs$(FSTYPE)    = ubifs
appconfigs$(PATSIZE)   = 0x400000
appconfigs$(MOUNTTG)  = /appconfigs
appconfigs$(MOUNTPT)  = ubi0:appconfigs
appconfigs$(OPTIONS)   = rw
appconfigs$(OTABLK)    = /dev/ubi0_3
```

不难发现，虽然分区表给rootfs中分配的了98M，但用df –h 看到只有85.6M，这是因为一部分用于分区了，就好像我们买了32G的U盘，其实真实的内存确实少了很多。

分区	大小
cis	0x40000=256K
ipl	0x20000=128K
ipl_cust	0x20000=128K
uboot	0x40000=256K
key_cust	0x20000=128K
logo	0x60000=384K
kernel	0x500000=5M
rootfs	0x6200000=98M
miservice	0xA00000=10M
customer	0x6500000=101M
appconfigs	0x400000=4M

我们可以对其进行修改，来修改分区的大小，这里以rootfs为例，这边把98M–>88M，缩小10M，也就是修改为0x5800000。修改后，我们看到我们开发板的内rootfs将近缩小了10M



```
/ # df -h
Filesystem                Size      Used Available Use% Mounted on
ubi:rootfs              170.7M      2.7M    168.0M   2% /
devtmpfs                 22.4M         0     22.4M   0% /dev
tmpfs                    22.4M         0     22.4M   0% /tmp
var                      22.4M         0     22.4M   0% /var
vendor                   22.4M         0     22.4M   0% /vendor
mdev                     22.4M         0     22.4M   0% /dev
ubi0:miservice            7.5M      5.4M      2.1M  71% /config
ubi0:customer             7.5M    160.0K      7.3M   2% /customer
ubi0:appconfigs           2.0M     24.0K      2.0M   1% /appconfigs
/ #
```

文件系统只读问题

系统起来后，我想创建一个文件，发现失败了，原因是文件系统是只读的：

```
/ # ls
appconfigs  customer    home        mnt         sys         var
bin         dev         lib         proc        tmp         vendor
config      etc         linuxrc     sbin        usr
/ # touch 123
touch: 123: Read-only file system
/ #
```

我们可以在project/image/configs/i2m/spinand.ubifs.p2.partition.config_256M修改为

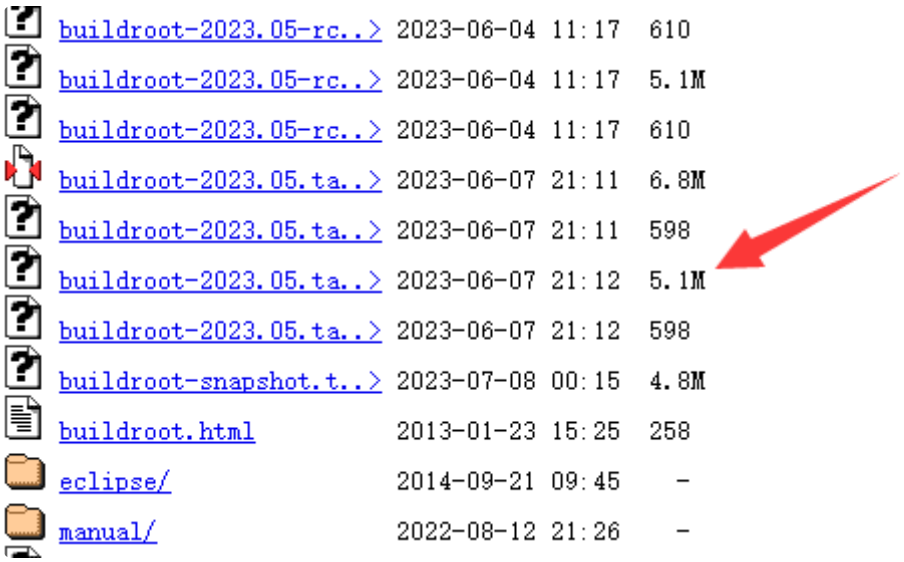
```
rootfs$(RESOUCE)   = $(OUTPUTDIR)/rootfs
rootfs$(FSTYPE)    = ubifs
rootfs$(PATSIZE)   = 0x5800000
rootfs$(BOOTENV)   = console=ttyS0,115200 ubi.mtd=UBI,2048 root=ubi:rootfs ro rootfstype=ubifs init=/linuxrc rootwait=1
```

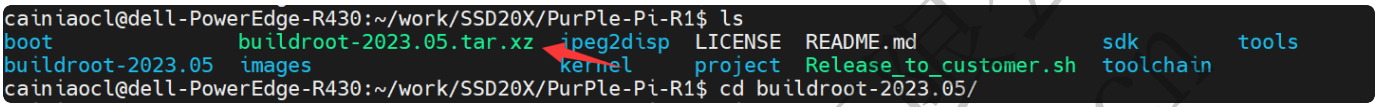
如果我把ro改成rw，应该就可以把rootfs设置为可读可写属性了：

```
SigmaStar # setenv bootargs console=ttyS0,115200 ubi.mtd=UBI,2048 root=ubi:rootfs rw rootfstype=ubifs init=/linuxrc
miu=0,sz=0x300000,max_start_off=0x3300000,max_end_off=0x3600000 mtdparts=nand0:384k@1280k(IPL0),384k(IPL1),384k(IPL
500000(RECOVERY),-(UBI)
SigmaStar # saveenv
Saving Environment to NAND...
Erasing NAND...
Erasing at 0x440000 -- 100% complete.
Writing to NAND... OK
SigmaStar #
SigmaStar #
```

buildroot文件系统定制

我们可以使用buildroot去定制一些我们所需要的功能，如添加wifi功能，wifi自启动连接，添加界面账号密码等功能。首先去我们的官网下载我们的buildroothttps://buildroot.org/downloads/





解压并进入buildroot进行配置

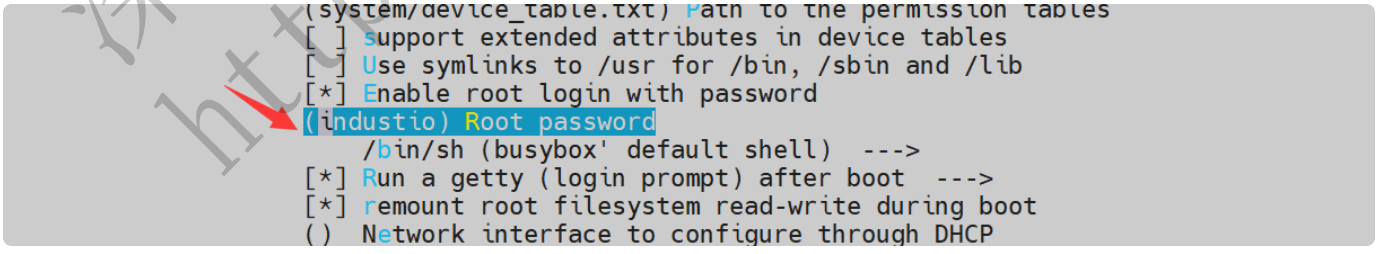
```
vi buildroot-2023.05
```

我么们根据ssd202来进行配置：

```
ARCH=arm make menuconfig
```

```
| |Target options  --->
| |      Target Architecture (ARM (little endian))  --->
| |      Target Architecture Variant (cortex-A7)  --->
| |      Floating point strategy (NEON)  --->

| |Toolchain  --->
| |          Toolchain type (External toolchain)  --->

| |          *** Toolchain External Options ***

| |          Toolchain (Custom toolchain)  --->
| |          (/home/cainiaocl/work/SSD20X/PurPle-Pi-R1/toolchain/gcc-arm-8.2-2018.08-x86_64-arm-linux-gnueabihf/
| |          (arm-linux-gnueabihf) Toolchain prefix

| |          External toolchain gcc version (8.x)  --->

| |          External toolchain kernel headers series (4.18.x)  --->

| |          External toolchain C library (glibc)  --->

| |        [*] Toolchain has SSP support? (NEW)

| |        [*]   Toolchain has SSP strong support? (NEW)

| |        [*] Toolchain has RPC support? (NEW)

| |        [*] Toolchain has C++ support?

```

上面是基础配置，如果需要配置登录的密码设置，我们进入System configuration在子菜单中，找到Enable root login with password选项选中，然后我们在 root password上填写我们的密码即可。



等编译完成之后，他会把编译好的rootfs.tar放到output/images/下。

```
cainiaocl@dell-PowerEdge-R430:~/work/SSD20X/PurPle-Pi-R1/buildroot-2023.05$ cd output/images/
cainiaocl@dell-PowerEdge-R430:~/work/SSD20X/PurPle-Pi-R1/buildroot-2023.05/output/images$ ls
rootfs.tar
```

，现在我们把buildroot生成的rootfs.tar替换我们的project/image/rootfs/rootfs.tar.gz即可.但这里需要注意的是，buildroot生成的rootfs.tar解压后是散包，没有套一层rootfs目录，但是我们项目下的rootfs.tar.gz却是由套一层rootfs，所以我们先把buildroot生成的rootfs.tar套一层目录：

Plain Text 复制代码

```
mkdir rootfs
tar -xvf rootfs.tar -C rootfs/
tar -cvf rootfs.tar.gz ./rootfs
```

然后再替换我们项目想的rootfs.tar.gz:

Plain Text 复制代码

```
cp rootfs.tar.gz  ../../../project/image/rootfs/
```

替换完成后，我们编译一下固件

Plain Text 复制代码

```
./Release_to_customer.sh -f nand -p ssd202 -m 256
```

快速启动模式fastboot

我们这款ssd202的开发板，已经有配置快速启动模式的的脚本了

Plain Text 复制代码

```
vi Release_to_customer.sh
```

```
while getopts "f:p:q:o:m:" opt; do
  case $opt in
    f)
      flashtype=$OPTARG
      ;;
    p)
      chip=$OPTARG
      ;;
    q)
      fastboot=$OPTARG
      ;;
    m)
      FLASH_SIZE=$OPTARG
      ;;
    \?)
      echo "Invalid option: -$OPTARG" >&2
      ;;
  esac
done
```

```
#build project
cd ${RELEASEDIR}/project
if [ "${flashtype}" = "nor" ]; then
        if [ "${fastboot}" = "fastboot" ]; then
                echo test fastboot
                ./setup_config.sh ./configs/nvr/i2m/8.2.1/nor.glibc-ramfs.011a.64
        else
                if [ "${chip}" = "ssd201" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/nor.glibc-squashfs.011a.64
                fi
                if [ "${chip}" = "ssd202" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/nor.glibc-squashfs.011a.128
                fi
        fi
else
        if [ "${fastboot}" = "fastboot" ]; then
                if [ "${chip}" = "ssd201" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.ram-glibc-squashfs.011a.64
                elif [ "${chip}" = "ssd202" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.ram-glibc-squashfs.011a.128
                fi
        else
                if [ "${chip}" = "ssd201" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.glibc.011a.64
                fi
                if [ "${chip}" = "ssd202" ]; then
                        ./setup_config.sh ./configs/nvr/i2m/8.2.1/spinand.glibc.011a.128
                fi
        fi

fi
```

我们只需要在运行脚本的时候添加：

Plain Text 复制代码

```
# ./Release_to_customer.sh -f nand -p ssd202 -q fastboot -m 256
```

同时，切换fastnoots需要修改下：

Plain Text 复制代码

```
vi project/image/configs/i2m/rootfs_fastboot.mk
```

```
echo "busybox telnetd&" >> $(OUTPUTDIR)/rootfs/etc/profile
#echo \/customer\/bin\/zkgui \& >> $(OUTPUTDIR)/rootfs/etc/profile;
#echo sleep 8 >> $(OUTPUTDIR)/rootfs/etc/profile;
#echo /customer/demo.sh >> $(OUTPUTDIR)/rootfs/etc/profile;
```

更新启动后，可以看到启动过程跳过了 uboot。

关闭网络功能，也能加快开机时间：

Plain Text 复制代码

```
setenv autoestar 0
saveenv
```

LCD配置

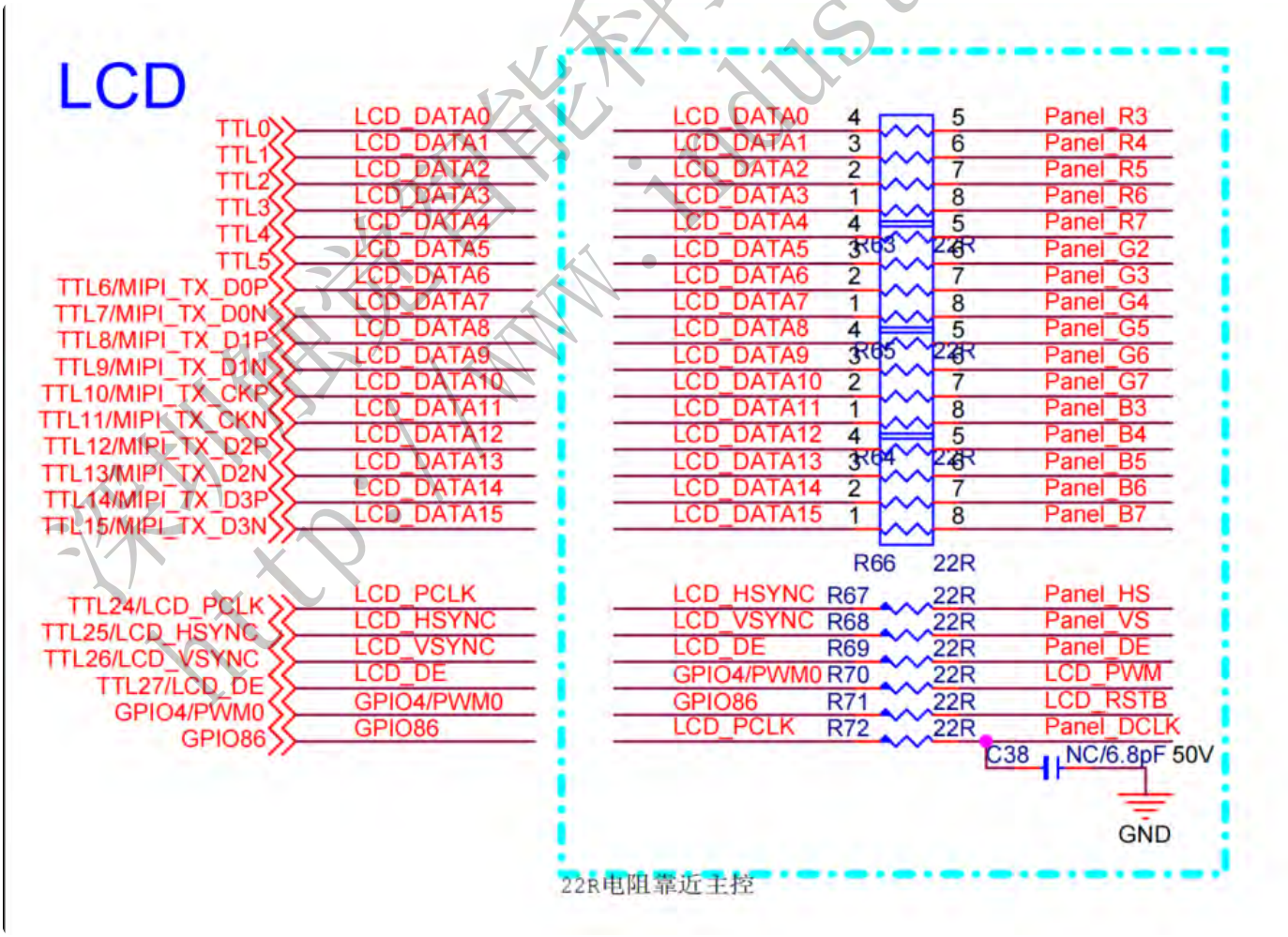
假设我们配置的屏幕信息是：720x1280，60fps、RGB888像素格式、4lane

kernel配置(dts配置)

点屏流程

dts的配置

我们需要配置MIPI屏幕，从原理图可以看出我们使用到的引脚是TTL1~15,且用到的数据时钟线是TTL6~15也就是D0P/N、D1P/N、D2P/N、D3P/N、CKP/N，这4个lane和1clk：



根据原理图可以看出，我们需要把屏幕配置成MIPI_MODE_1

Power Domain default 3.3V	Ball Pin Name	GPIO index	default value after reset	default Rpu/Rpu	5V tolerance	Driving	Interrupt Available edge, rising/falling	reg_tx_mipi_mode
								1
								10pin
AVDD_NODIE	PAD_UART1_RX	49	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
AVDD_NODIE	PAD_UART1_TX	50	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_FUART_RX	15	input, pull-up	Rpu, 90K ohm	Yes	>4mA	Yes	
VDDP_1	PAD_FUART_TX	16	input, pull-up	Rpu, 90K ohm	Yes	>4mA	Yes	
VDDP_1	PAD_FUART_CTS	17	input, pull-down	Rpd, 103K ohm	Yes	>4mA	Yes	
VDDP_1	PAD_FUART_RTS	18	input, pull-down	Rpd, 103K ohm	Yes	>4mA	Yes	
VDDP_1	PAD_TTL0	19	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL1	20	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL2	21	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL3	22	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL4	23	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL5	24	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL6	25	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_P_CH0
VDDP_1	PAD_TTL7	26	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_N_CH0
VDDP_1	PAD_TTL8	27	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_P_CH1
VDDP_1	PAD_TTL9	28	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_N_CH1
VDDP_1	PAD_TTL10	29	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_P_CH2
VDDP_1	PAD_TTL11	30	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_N_CH2
VDDP_1	PAD_TTL12	31	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_P_CH3
VDDP_1	PAD_TTL13	32	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_N_CH3
VDDP_1	PAD_TTL14	33	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_P_CH4
VDDP_1	PAD_TTL15	34	input, pull-up	Rpu, 90K ohm	No	DRV=0 ->4mA DRV=1 ->8mA	Yes	MIPI_TX_N_CH4
VDDP_1	PAD_TTL16	35	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL17	36	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL18	37	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL19	38	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL20	39	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
VDDP_1	PAD_TTL21	40	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	
						DRV=0 ->4mA		

需要把引脚配置成MIPI模式：

Plain Text 复制代码

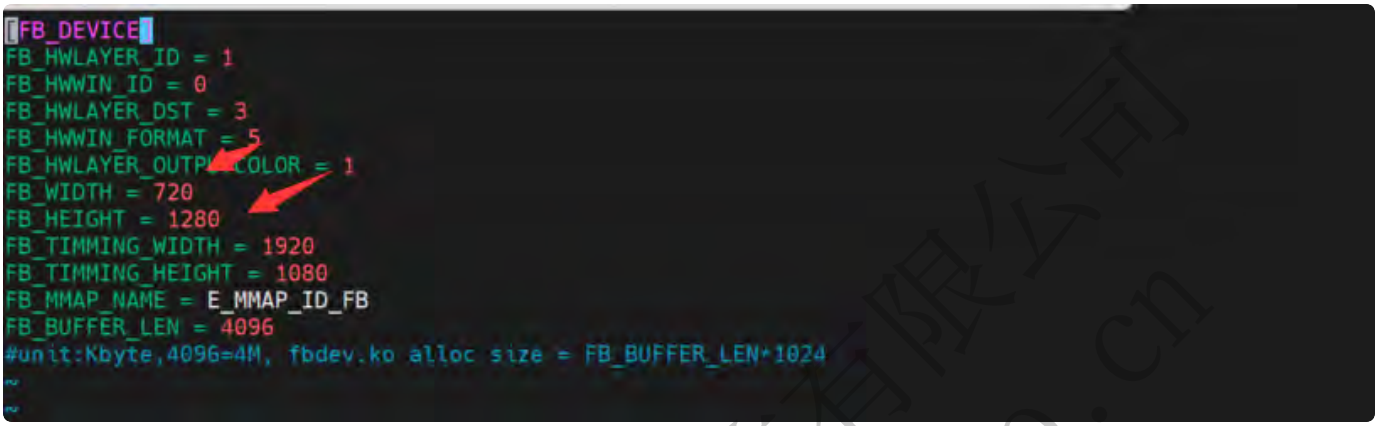
```
1  vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```

```
<PAD_TTL6          PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_P_CH0 >,
<PAD_TTL7          PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_N_CH0>,
<PAD_TTL8          PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_P_CH1>,
<PAD_TTL9          PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_N_CH1 >,
<PAD_TTL10         PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_P_CH2 >,
<PAD_TTL11         PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_N_CH2 >,
<PAD_TTL12         PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_P_CH3 >,
<PAD_TTL13         PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_N_CH3 >,
<PAD_TTL14         PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_P_CH4 >,
<PAD_TTL15         PINMUX_FOR_TX_MIPI_MODE_1  MDRV_PUSE_TX_MIPI_N_CH4 >,
```

project的配置

修改屏幕分辨率为720x1280

```
project/board/i2m/SSC011A-S01A/config/fbdev.ini
```



自启动的话，把jpeg2disp的enable，并且把disp_init给disable

```
vi project/release/customer_tailor/nvr_i2m_display_glibc_tailor.mk
```

jpeg2disp的配置

配置屏参文件

1）我们配置屏参文件，我们需要知道屏幕的这些信息：

1. 分辨率：720x1280

2. HSyncWidth：30 ； HSyncBackPorch：30 ； HSyncFrontPorch：60

3. VSyncWidth：4 ； VSyncBackPorch：12 ； HVyncFrontPorch：18

4. 帧：60HZ

5. date-lane：4

6. 像素格式：rgb888

2）打开我们的屏参文件，根据我们得到分辨率以及屏幕时钟等这些信息，直接这里替换：

```
5
6   #define HPW (8)
7   #define HBP (24)
8   #define HFP (16)
9   #define VPW (68)
10  #define VBP (120)
11  #define VFP (88)
12
13  #define FPS (60)
14  #define HDA (720)
15  #define VDA (1280)
16
```

直接在这里替换

3）像素时钟和数据lane直接替换，

```
386
387   E_MI_PNL_MIPI_DSI_LANE_4,      // MIPnlMipiDsiLaneMode_e enLaneNum;
388   E_MI_PNL_MIPI_DSI_RGB888,      // MIPnlMipiDsiFormat_e enFormat;
389   E_MI_PNL_MIPI_DSI_SYNC_PULSE,  // MIPnlMipiDsiCtrlMode_e enCtrl;
390
```

4）如果在你的数据lane中，屏幕接口上的DxP和DxN是跟开发板上的DxP和DxN是相反的话，我们修改这个为2,2,2,2,2,

```
386
387       E_MI_PNL_MIPI_DSI_LANE_4,      // MIPnlMipiDsiLaneMode_e enLaneNum;
388       E_MI_PNL_MIPI_DSI_RGB888,      // MIPnlMipiDsiFormat_e enFormat;
389       E_MI_PNL_MIPI_DSI_SYNC_PULSE,  // MIPnlMipiDsiCtrlMode_e enCtrl;
390
391       MYTEST_800x1280,
392       sizeof(MYTEST_800x1280),
393       1, 0x01AF, 0x0159, 0x80D2, 7,
394       0,0,0,0,0,
395   };
396
```

m_ucPolCh0	Chn0极性 0：default 1：positive 2：negative
m_ucPolCh1	Chn1极性 0：default 1：positive 2：negative
m_ucPolCh2	Chn2极性 0：default 1：positive 2：negative
m_ucPolCh3	Chn3极性 0：default 1：positive 2：negative
m_ucPolCh4	Chn4极性 0：default 1：positive 2：negative

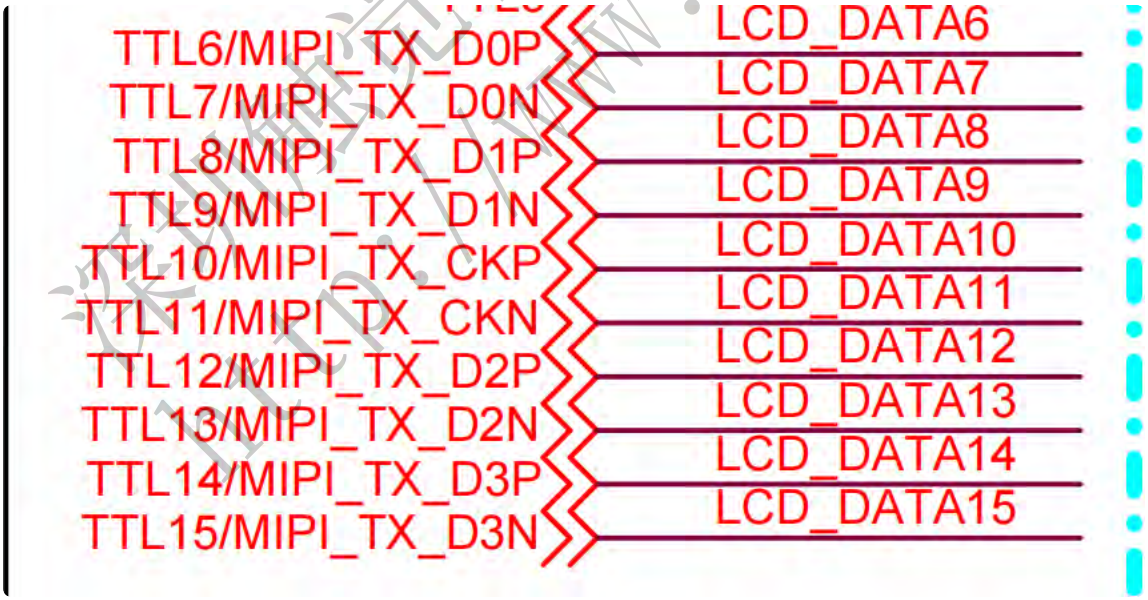
5）配置MI_PANEL_ChannelSwapType_e

在我们结构体中我们通道的顺序就是这样，从上到下clk、lane3、Lane2、Lane1、Lane0。

Plain Text 复制代码

```
1.  typedef struct
2.  {
5.      MI_PANEL_ChannelSwapType_e eCh0;   //--> CLK 栏位(CKN/P)
6.      MI_PANEL_ChannelSwapType_e eCh1;   //--> Lane3栏位(D3N/P)
7.      MI_PANEL_ChannelSwapType_e eCh2;   //--> Lane2栏位(D2N/P)
8.      MI_PANEL_ChannelSwapType_e eCh3;   //--> Lane1栏位(D1N/P)
9.      MI_PANEL_ChannelSwapType_e eCh4;   //--> Lane0栏位(D0N/P)
10. }MI_PANEL_ParamConfig_t;
```

需要参考实际的硬件连接配置：



我们TTL对应的枚举情况：

Plain Text 复制代码

```
芯片内部默认情况下对应的pin是：
Lane0对应的是PAD_TTL6/7 → E_MI_PNL_CH_SWAP_0 （对应枚举值是0）
Lane1对应的是PAD_TTL8/9 → E_MI_PNL_CH_SWAP_1 （对应枚举值是1）
Lane2对应的是PAD_TTL12/13 → E_MI_PNL_CH_SWAP_3 （对应枚举值是3）
Lane3对应的是PAD_TTL14/15 → E_MI_PNL_CH_SWAP_4 （对应枚举值是4）
Clk对应的是PAD_TTL10/11 → E_MI_PNL_CH_SWAP_2 （对应枚举值是2）
```

Plain Text 复制代码

```
m_ucCl        Clk  lane  selection(default:2) 0:select chn0 1:select chn1
2:select chn2 3:select chn3 4:select chn4
m_ucDataLane0 data lane0 selection(default:4) 0:select chn0 1:select chn1
2:select chn2 3:select chn3 4:select chn4
m_ucDataLane1 data lane1 selection(default:3) 0:select chn0 1:select chn1
2:select chn2 3:select chn3 4:select chn4
m_ucDataLane2 data lane2 selection(default:1) 0:select chn0 1:select chn1
2:select chn2 3:select chn3 4:select chn4
m_ucDataLane3 data lane3 selection(default:0) 0:select chn0 1:select chn1
2:select chn2 3:select chn3 4:select chn4
```

所以，从原理图看：

我们的CKN/P对应的是TTL10和TTL11，对应的是E_MI_PNL_CH_SWAP_2，所以第一个枚举值是2，

我们的D3N/P对应的是TTL14和TTL15，对应的是E_MI_PNL_CH_SWAP_4，所以第一个枚举值是4

我们的D2N/P对应的是TTL12和TTL13，对应的是E_MI_PNL_CH_SWAP_3，所以第一个枚举值是3，

我们的D1N/P对应的是TTL8和TTL9，对应的是E_MI_PNL_CH_SWAP_1，所以第一个枚举值是1，

我们的D0N/P对应的是TTL6和TTL7，对应的是E_MI_PNL_CH_SWAP_0，所以第一个枚举值是0，

Plain Text 复制代码

```
(MI_PANEL_ChannelSwapType_e)2,
(MI_PANEL_ChannelSwapType_e)4,
(MI_PANEL_ChannelSwapType_e)3,
(MI_PANEL_ChannelSwapType_e)1,
(MI_PANEL_ChannelSwapType_e)0,
```

参考屏参文件如下：

U5505HA-WA0-720x1280.h Plain Text 复制代码

```
#include "mi_panel_datatype.h"

#define FLAG_DELAY            0xFE
#define FLAG_END_OF_TABLE     0xFF   // END OF REGISTERS MARKER

#define HPW (8)
#define HBP (24)
#define HFP (16)
#define VPW (68)
#define VBP (120)
#define VFP (88)

#define FPS (60)
#define HDA (720)
#define VDA (1280)


MI_PANEL_ParamConfig_t stPanelParam =
{
    "WT070BM24_800x1280_60", // const char *m_pPanelName;
///<  PanelName
    0, //MS_U8 m_bPanelDither :1;                ///<  PANEL_DITHER, kee
p the setting
    E_MI_PNL_LINK_MIPI_DSI, //MHAL_DISP_ApiPnlLinkType_e m_ePanelLinkTyp
e   :4;  ///<  PANEL_LINK

    ////////////////////////////////////////////
    // Board related setting
    ////////////////////////////////////////////
    1,  //MS_U8 m_bPanelDualPort  :1;             ///<  VOP_21[8], MOD_4
A[1],    PANEL_DUAL_PORT, refer to m_bPanelDoubleClk
    0,  //MS_U8 m_bPanelSwapPort  :1;             ///<  MOD_4A[0],
         PANEL_SWAP_PORT, refer to "LVDS output app note" A/B channel swa
p
    0,  //MS_U8 m_bPanelSwapOdd_ML    :1;         ///<  PANEL_SWAP_ODD_M
L
    0,  //MS_U8 m_bPanelSwapEven_ML   :1;         ///<  PANEL_SWAP_EVEN_
ML
    0,  //MS_U8 m_bPanelSwapOdd_RB    :1;         ///<  PANEL_SWAP_ODD_R
B
    0,  //MS_U8 m_bPanelSwapEven_RB   :1;         ///<  PANEL_SWAP_EVEN_
RB

    0,  //MS_U8 m_bPanelSwapLVDS_POL  :1;         ///<  MOD_40[5], PANEL
_SWAP_LVDS_POL, for differential P/N swap
```

```
    0,  //MS_U8 m_bPanelSwapLVDS_CH   :1;          ///<  MOD_40[6], PANEL_SWAP_LVDS_CH, for pair swap
    0,  //MS_U8 m_bPanelPDP10BIT      :1;          ///<  MOD_40[3], PANEL_PDP_10BIT ,for pair swap
    1,  //MS_U8 m_bPanelLVDS_TI_MODE  :1;          ///<  MOD_40[2], PANEL_LVDS_TI_MODE, refer to "LVDS output app note"

    //////////////////////////////////////////////
    // For TTL Only
    //////////////////////////////////////////////
    0,  //MS_U8 m_ucPanelDCLKDelay;                ///<  PANEL_DCLK_DELAY
    0,  //MS_U8 m_bPanelInvDCLK   :1;              ///<  MOD_4A[4],
            PANEL_INV_DCLK
    0,  //MS_U8 m_bPanelInvDE     :1;              ///<  MOD_4A[2],
            PANEL_INV_DE
    0,  //MS_U8 m_bPanelInvHSync  :1;              ///<  MOD_4A[12],
            PANEL_INV_HSYNC
    0,  //MS_U8 m_bPanelInvVSync  :1;              ///<  MOD_4A[3],
            PANEL_INV_VSYNC

    //////////////////////////////////////////////
    // Output driving current setting
    //////////////////////////////////////////////
    // driving current setting (0x00=4mA, 0x01=6mA, 0x02=8mA, 0x03=12mA)
    1,  //MS_U8 m_ucPanelDCKLCurrent;              ///<  define PANEL_DCLK_CURRENT
    1,  //MS_U8 m_ucPanelDECurrent;                ///<  define PANEL_DE_CURRENT
    1,  //MS_U8 m_ucPanelODDDataCurrent;           ///<  define PANEL_ODD_DATA_CURRENT
    1,  //MS_U8 m_ucPanelEvenDataCurrent;          ///<  define PANEL_EVEN_DATA_CURRENT

    //////////////////////////////////////////////
    // panel on/off timing
    //////////////////////////////////////////////
    30,  //MS_U16 m_wPanelOnTiming1;               ///<  time between panel & data while turn on power
    400,  //MS_U16 m_wPanelOnTiming2;              ///<  time between data & back light while turn on power
    80,  //MS_U16 m_wPanelOffTiming1;              ///<  time between back light & data while turn off power
    30,  //MS_U16 m_wPanelOffTiming2;              ///<  time between data & panel while turn off power

    //////////////////////////////////////////////
    // panel timing spec.
    //////////////////////////////////////////////
```

```
    // sync related
    HPW,  //MS_U8 m_ucPanelHSyncWidth;              ///<  VOP_01[7:0], PANEL_HSYNC_WIDTH
    HBP,  //MS_U8 m_ucPanelHSyncBackPorch;          ///<  PANEL_HSYNC_BACK_PORCH, no register setting, provide value for query only

                                                   ///<  not support Manuel VSync Start/End now
                                                   ///<  VOP_02[10:0] VSync start = Vtt - VBackPorch - VSyncWidth
                                                   ///<  VOP_03[10:0] VSync end = Vtt - VBackPorch
    VPW,  //MS_U8 m_ucPanelVSyncWidth;              ///<  define PANEL_VSYNC_WIDTH
    VBP,  //MS_U8 m_ucPanelVBackPorch;              ///<  define PANEL_VSYNC_BACK_PORCH

    // DE related
    (HPW+HBP),  //MS_U16 m_wPanelHStart;           ///<  VOP_04[11:0], PANEL_HSTART, DE H Start (PANEL_HSYNC_WIDTH + PANEL_HSYNC_BACK_PORCH)
    (VPW+VBP),  //MS_U16 m_wPanelVStart;                ///<  VOP_06[11:0], PANEL_VSTART, DE V Start
    HDA,  //MS_U16 m_wPanelWidth;                  ///< PANEL_WIDTH, DE width (VOP_05[11:0] = HEnd = HStart + Width - 1)
    VDA,  //MS_U16 m_wPanelHeight;                 ///< PANEL_HEIGHT, DE height (VOP_07[11:0], = Vend = VStart + Height - 1)

    // DClk related
    0,  //MS_U16 m_wPanelMaxHTotal;               ///<  PANEL_MAX_HTOTAL. Reserved for future using.
    (HDA+HPW+HBP+HFP),  //MS_U16 m_wPanelHTotal;                  ///< VOP_0C[11:0], PANEL_HTOTAL
    0,  //MS_U16 m_wPanelMinHTotal;               ///<  PANEL_MIN_HTOTAL. Reserved for future using.

    0,  //MS_U16 m_wPanelMaxVTotal;               ///<  PANEL_MAX_VTOTAL. Reserved for future using.
    (VDA+VPW+VBP+VFP),  //MS_U16 m_wPanelVTotal;                  ///< VOP_0D[11:0], PANEL_VTOTAL
    0,  //MS_U16 m_wPanelMinVTotal;               ///<  PANEL_MIN_VTOTAL. Reserved for future using.

    0,  //MS_U8 m_dwPanelMaxDCLK;                 ///<  PANEL_MAX_DCLK. Reserved for future using.
    ((unsigned long)(VDA+VPW+VBP+VFP)*(HDA+HPW+HBP+HFP)*FPS/1000000),  //MS_U8 m_dwPanelDCLK;                    ///<  LPLL_0F[23:0], PANEL_DCLK
        ,{0x3100_10[7:0], 0x3100_0F[15:0]}
```

```
    0,  //MS_U8 m_dwPanelMinDCLK;                  ///<  PANEL_MIN_DCLK.
Reserved for future using.
                                                 ///<  spread spectrum
    0,  //MS_U16 m_wSpreadSpectrumStep;           ///<  move to board de
fine, no use now.
    0,  //MS_U16 m_wSpreadSpectrumSpan;           ///<  move to board de
fine, no use now.

    160,  //MS_U8 m_ucDimmingCtl;                   ///<  Initial Dimmin
g Value
    255,  //MS_U8 m_ucMaxPWMVal;                    ///<  Max Dimming Va
lue
    80,  //MS_U8 m_ucMinPWMVal;                     ///<  Min Dimming Val
ue

    0,  //MS_U8 m_bPanelDeinterMode   :1;          ///<  define PANEL_DEI
NTER_MODE,  no use now
    E_MI_PNL_ASPECT_RATIO_WIDE,  //MHAL_DISP_PnlAspectRatio_e m_ucPanelAs
pectRatio; ///<  Panel Aspect Ratio, provide information to upper layer a
pplication for aspect ratio setting.
  /*
    *
    * Board related params
    *
    *  If a board ( like BD_MST064C_D01A_S ) swap LVDS TX polarity
    *    : This polarity swap value =
    *      (LVDS_PN_SWAP_H<<8) | LVDS_PN_SWAP_L from board define,
    *  Otherwise
    *    : The value shall set to 0.
    */
    0,  //MS_U16 m_u16LVDSTxSwapValue;
    E_MI_PNL_TI_8BIT_MODE,  //MHAL_DISP_ApiPnlTiBitMode_e m_ucTiBitMode;
                            ///< MOD_4B[1:0], refer to "LVDS output app note"
    E_MI_PNL_OUTPUT_8BIT_MODE,  //MHAL_DISP_ApiPnlOutPutFormatBitMode_e m
_ucOutputFormatBitMode;

    0,  //MS_U8 m_bPanelSwapOdd_RG    :1;          ///<  define PANEL_SWA
P_ODD_RG
    0,  //MS_U8 m_bPanelSwapEven_RG   :1;          ///<  define PANEL_SWA
P_EVEN_RG
    0,  //MS_U8 m_bPanelSwapOdd_GB    :1;          ///<  define PANEL_SWA
P_ODD_GB
    0,  //MS_U8 m_bPanelSwapEven_GB   :1;          ///<  define PANEL_SWA
P_EVEN_GB

    /**
    *  Others
    */
```

```
128         1,  //MS_U8 m_bPanelDoubleClk     :1;             ///<  LPLL_03[7], d
    efine Double Clock ,LVDS dual mode
129         0x001c848e,  //MS_U32 m_dwPanelMaxSET;                  ///<  defi
    ne PANEL_MAX_SET
130         0x0018eb59,  //MS_U32 m_dwPanelMinSET;                  ///<  defi
    ne PANEL_MIN_SET
131         E_MI_PNL_CHG_VTOTAL,  //MHAL_DISP_ApiPnlOutTimingMode_e m_ucOutTiming
```

jpegdisp的配置

我们要根据原理图，找出我们的LCD_RST和PWM_RST，这两个参数是背光和屏幕的引脚，通过使用该引脚才会顺利的点亮我们的屏幕。通过我们的原理图，我们可以看出对应的引脚号是LCD_RST—>GPIO86和PWM_RST—>GPIO4

所以，在我们的/home/cainiaocl/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/run.sh里配置我们的开机自启动屏幕：

```
echo 4 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio4/direction
echo 1 > /sys/class/gpio/gpio4/value

echo 86 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio86/direction
echo 1 > /sys/class/gpio/gpio86/value


chmod 777 logo_mge
chmod 777 logo.jpg


./logo_mge logo.jpg
```

我们如果想修改我们的生成图片的执行文件，我们就在：/home/cainiaocl/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/src/makefile下修改这个变量，更改我们的名称

```
STDLIBS:=  -lstdc++ -ldl  -lpthread -lm
LIBS:=-Wl,-Bdynamic ${MINIGUI_LIBS} ${PLAT_DEPENDENT_LIB_EX} ${PLAT_DEPENDENT_LIB} -Wl,-Bdynamic $(STDLIBS)
CFLAGS:=-rdynamic -funwind-tables -I. -I$(SDK_INCS) -I$(APPLICATION_PATH)/jpeg2disp/inc -I$(UAPI_INCS)
#-I$(PWD)/library/include -I$(PWD)/library/include/ctrl -I$(PWD)/library/include/libpng16/ -I/usr/local/include -I$(U
CC:=arm-linux-gnueabihf-gcc


LOGO_SRC:=$(APPLICATION_PATH)/jpeg2disp/src/logo.c  \
          $(APPLICATION_PATH)/jpeg2disp/src/sstardisp.c \
          $(APPLICATION_PATH)/jpeg2disp/src/bmp.c  \
          $(APPLICATION_PATH)/jpeg2disp/src/jpeg.c

LOGO:=logo_mge

.PHONY : app_src app_clean


app_src:
```

相对应的我们需要在/home/cainiaocl/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/image.mk下，修改对应的执行文件的名称，以及我们的图片名称

```
include $(APPLICATION_PATH)/jpeg2disp/src/makefile
app:app_src
ifeq ($(IMAGE_INSTALL_DIR),)
        @echo "directory of image is not defined"
        @exit 1
endif
        @cp -rf $(APPLICATION_PATH)/jpeg2disp/lib $(IMAGE_INSTALL_DIR)/customer
        @$(TOOLCHAIN_REL)strip --strip-unneeded $(IMAGE_INSTALL_DIR)/customer/lib/*
ifeq ($(PROJECT), xiangfan)
        @cp -rf $(APPLICATION_PATH)/jpeg2disp/res/xiangfan480x480.jpg $(IMAGE_INSTALL_DIR)/customer/logo.jpg
        @cat $(APPLICATION_PATH)/jpeg2disp/run_xiangfan.sh >> $(IMAGE_INSTALL_DIR)/customer/demo.sh
else
        @cp -rf $(APPLICATION_PATH)/jpeg2disp/res/logo720x1280.jpg $(IMAGE_INSTALL_DIR)/customer/logo.jpg
        @cat $(APPLICATION_PATH)/jpeg2disp/run.sh >> $(IMAGE_INSTALL_DIR)/customer/demo.sh
endif
        @cp -rf $(APPLICATION_PATH)/jpeg2disp/src/logo_mge $(IMAGE_INSTALL_DIR)/customer
        @$(TOOLCHAIN_REL)strip --strip-unneeded $(IMAGE_INSTALL_DIR)/customer/logo_mge
        @make app_clean
```

我们的图片文件应该放在：/home/cainiaocl/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/res

```
cainiaocl@dell-PowerEdge-R430:~/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/res$ ls
logo1024x600.jpeg  logo240x320.jpg  logo480x480.jpg  logo720x1280.jpg  logo800x1280.jpg  logo.jpg
logo1024x600.jpg   logo320x240.jpg  logo720x1280-1.jpg  logo720x720.jpg  logo800x480.jpg  xiangfan480x480.jpg
cainiaocl@dell-PowerEdge-R430:~/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/res$
```

同时，我们把屏参文件放到该目录下：/home/cainiaocl/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/src

```
cainiaocl@dell-PowerEdge-R430:~/work/SSD20X/PurPle-Pi-R1/sdk/verify/application/jpeg2disp/src$ ls
AA121SL01-TTL-800x600.h          FRD720x720_MIPI.h                logo_mge                         SH8601A0_480x480_60.h            U5010HA-WA_1280x720_MIPI.h
blitutil.c                       FRD720x720_MIPI_o.h              logo_QS                          SH8601A_ilane_454x454_MIPI_cp.h  U5505HA-WA0-720x1280.h
blitutil.h                       GVS_3.97_BOE_720P.TXT            logo_rsd                         SH8601A_ilane_454x454_MIPI.h     U5505HA-WA0-720x1280-new.h
bmp.c                            HD69001C40_280x1424_MIPI.h       logo_xiangfan.c                  sstardisp_2D07.c                 update_disp_init.txt
bmp.h                            IWXIM31B_800x1280_MIPI.h         LQ035_320x240b.h                 sstardisp.c                      verify_gfx.c
CC021I40R2_480x480.h             jpeg.c                           makefile                         sstardisp.h                      verify_gfx.h
CC0702I50R_1024x600.h            jpeg.h                           MC10132007_800x1280.h            sstardisp_xiangfan.c             verify_gfx_type.h
conf.sh                          JZH21BS818C.h                    mypng.c                          ST7701S_480x480.h                VS040BST07V0_720x720.h
disp_init                        KWH070KQ48_720x1280_MIPI_bk.h    mypng.h                          ST7703_H047A3_720x1280_MIPI.h    WT021B4000_480x480.h
disp_init.c                      KWH070KQ48_720x1280_MIPI.h       NV3047_480x272.h                 TG040H02_720x720_MIPI.h          WT028QVTMA_240x320.h
EK79007_1024x600_MIPI.h          logo                             ReadMe.txt                       tianhai_1200x1920.h              WT070BM24_800x1280_MIPI.h
EQT700BKJ004P_1024x600_MIPI.h    logo_bk.c                        'reference driver'               TS_P0700130A_800x1280_MIPI.h     Z21002_480x480_MIPI.h
FPGA_800x480_60.h                logo.c                           SAT070AT50_800x480.h             tuoren_280x1424.h                ZH_1024x600.h
FRD720x720_MIPI1.h               logo.jpg                         SAT070CP50_1024x600.h            tuoren_280x1424.zip
```

并srcsstardisp.c中把自己屏参文件添加上去同时屏蔽其他的屏参文件

```
#include "mi_disp_datatype.h"
#include "mi_disp.h"


#define UI_1024_600 0
#define USE_MIPI    1

#include "U5505HA-WA0-720x1280-new.h"

//#define LOCAL_VIDEO_X  100
//#define LOCAL_VIDEO_Y  60
#define LOCAL_VIDEO_W  720
#define LOCAL_VIDEO_H  1280
#define LOCAL_VIDEO_X  0
#define LOCAL_VIDEO_Y  0
```

重新编译一下，即可

```
./Release_to_customer.sh -f nand -p ssd201
```

常见问题

1）报错提示找不到-ljpeg，即libjpeg.so，但是在找到libjpeg.so.7：

报错提示找不到-ljpeg，即libjpeg.so，但是在找到libjpeg.so.7：

```
# ls sdk/verify/application/jpeg2disp/lib
```

```
industio@industio:~/ssd20x$ ls sdk/verify/application/jpeg2disp/lib
libjpeg.so.7  libjpeg.so.7.0.0  libz.so.1  libz.so.1.2.2
industio@industio:~/ssd20x$
industio@industio:~/ssd20x$
industio@industio:~/ssd20x$
```

做一个软链接即可：

```
# cd sdk/verify/application/jpeg2disp/lib
# ln -s libjpeg.so.7 libjpeg.so
# ln -s libz.so.1 libz.so
```

重新编译并更新固件：

```
# cd -
# ./Release_to_customer.sh -f nand -p ssd202 -m 256
```

2）屏幕不亮的解决办法：

第一，先检查背光是否有亮，没有亮就先打开背光，如果背光亮了没有显示图片，就把屏幕的LCD_RST引脚给电，顺便测量LCD_RST和PWM_RST是否有电压能被控制，如果都没有问题，就查看自己的dts的引脚模式是否有配置好。

第二，如果都还是不亮，查看是否有上电时序，有些屏幕有特殊要求，需要上电时序才会复位正确才能亮屏

第三，在看看 检查开机logo是否有关掉。

第四，可能你的转接线太长导致数据传输不过来也有可能。

第五、分辨率HSyncWidth，HSyncBackPorch，HSyncFrontPorch，VSyncWidth，VSyncBackPorch，HVyncFrontPorch，帧率，date-lane，像素格式是否正确。

第六、如果都还是有问题，去厂家把屏参对一下是否正确，且再用自检参数试一下，如果还是没反应，可能出现硬件的问题去排查一下，根据原理图对底板跟转接板的元器件进行检测，是否有电压。

第七、或者你的内核没更新，你使用uname -a 查看一下，最新的编译内核的时间，如果不是最信的，可能你编译固件的时候出现了内核错误，查看下日志，可能你忽略了。

第八，用示波器测试一下。

第九、如果有屏幕抖动，可能因为延迟或者帧率，适当修改即可。

第十、在调试tp的时候，我们需要通过测量中断脚也就是inr脚，在我们按压屏幕的时候，电压值会发生变化。或者通过测量各个引脚，是否通电，也就是测量电阻值。

第十一、我们使用bootlogo，切换logo，存在短时间的黑屏：echo 1 > /sys/class/mstar/mdisp/bootlogo。

第十二、使用该命令查看屏参，看我们的屏参是否没有写好：cat proc/mi_modules/mi_panel/mi_panel0

第十三、我们进入该文件查看我们的分辨率是否正确：：vi config/fbdev.ini。

第十三、如果涉及纳秒级别的时序要求，需要在uboot，或者kernel上做点屏时序处理，在开发板上，sleep精确度不高，存在误差，不能实现这个操作。

bootlogo配置

bootlogo配置相对简单，如果我们的屏幕已经点亮，并可以显示图片的话吗，我们需要在原来屏参的基础上进行修改为适配bootlogo的屏参文件，变动不大只是修改下对应的结构体：

Plain Text 复制代码

```
// #include "mi_panel_datatype.h"

#ifndef SH8601A_454x454_MIPI_BOOTLOGO_H
#define SH8601A_454x454_BOOTLOGO_H

#define FLAG_DELAY            0xFE
#define FLAG_END_OF_TABLE     0xFF   // END OF REGISTERS MARKER

#define HPW (30)
#define HBP (30)
#define HFP (60)
#define VPW (4)
#define VBP (12)
#define VFP (18)

#define FPS (60)
#define HDA (720)
#define VDA (1280)


MhalPnlParamConfig_t stPanelParam_U5505HA_WA0_720x1280 =
{
    "U5505HA_WA0_720x1280", // const char *m_pPanelName;
///<  PanelName
#if !defined (__aarch64__)
    0,
#endif

  0, //MS_U8 m_bPanelDither :1;                ///<  PANEL_DITHER, keep
the setting
    E_MHAL_PNL_LINK_MIPI_DSI, //MHAL_DISP_ApiPnlLinkType_e m_ePanelLinkTy
pe   :4;  ///<  PANEL_LINK

    ////////////////////////////////////////////
    // Board related setting
    ////////////////////////////////////////////
    1,  //MS_U8 m_bPanelDualPort  :1;             ///<  VOP_21[8], MOD_4
A[1],     PANEL_DUAL_PORT, refer to m_bPanelDoubleClk
    0,  //MS_U8 m_bPanelSwapPort  :1;             ///<  MOD_4A[0],
          PANEL_SWAP_PORT, refer to "LVDS output app note" A/B channel swa
p
    0,  //MS_U8 m_bPanelSwapOdd_ML    :1;         ///<  PANEL_SWAP_ODD_M
L
    0,  //MS_U8 m_bPanelSwapEven_ML   :1;         ///<  PANEL_SWAP_EVEN_
ML
```

```
38      0,  //MS_U8 m_bPanelSwapOdd_RB    :1;          ///<  PANEL_SWAP_ODD_R
    B
39      0,  //MS_U8 m_bPanelSwapEven_RB   :1;          ///<  PANEL_SWAP_EVEN_
    RB
40
41      0,  //MS_U8 m_bPanelSwapLVDS_POL  :1;          ///<  MOD_40[5], PANEL
    _SWAP_LVDS_POL, for differential P/N swap
42      0,  //MS_U8 m_bPanelSwapLVDS_CH   :1;          ///<  MOD_40[6], PANEL
    _SWAP_LVDS_CH, for pair swap
43      0,  //MS_U8 m_bPanelPDP10BIT      :1;          ///<  MOD_40[3], PANEL
    _PDP_10BIT ,for pair swap
44      1,  //MS_U8 m_bPanelLVDS_TI_MODE  :1;          ///<  MOD_40[2], PANEL
    _LVDS_TI_MODE, refer to "LVDS output app note"
45
46
47      //////////////////////////////////////////////
48      // For TTL Only
49      //////////////////////////////////////////////
50      0,  //MS_U8 m_ucPanelDCLKDelay;                ///<  PANEL_DCLK_DELAY
        0,  //MS_U8 m_bPanelInvDCLK   :1;              ///<  MOD_4A[4],
                PANEL_INV_DCLK
51      0,  //MS_U8 m_bPanelInvDE     :1;              ///<  MOD_4A[2],
                PANEL_INV_DE
52      0,  //MS_U8 m_bPanelInvHSync  :1;              ///<  MOD_4A[12],
                PANEL_INV_HSYNC
53      0,  //MS_U8 m_bPanelInvVSync  :1;              ///<  MOD_4A[3],
                PANEL_INV_VSYNC
54
55
56      //////////////////////////////////////////////
57      // Output driving current setting
58      //////////////////////////////////////////////
59      // driving current setting (0x00=4mA, 0x01=6mA, 0x02=8mA, 0x03=12mA)
        1,  //MS_U8 m_ucPanelDCKLCurrent;              ///<  define PANEL_DCL
    K_CURRENT
60      1,  //MS_U8 m_ucPanelDECurrent;                ///<  define PANEL_DE_
    CURRENT
61      1,  //MS_U8 m_ucPanelODDDataCurrent;           ///<  define PANEL_ODD
    _DATA_CURRENT
62      1,  //MS_U8 m_ucPanelEvenDataCurrent;          ///<  define PANEL_EVE
    N_DATA_CURRENT
63
64
65      //////////////////////////////////////////////
66      // panel on/off timing
67      //////////////////////////////////////////////
        30,  //MS_U16 m_wPanelOnTiming1;                ///<  time between pa
    nel & data while turn on power
68      400,  //MS_U16 m_wPanelOnTiming2;               ///<  time between d
    ata & back light while turn on power
69
```

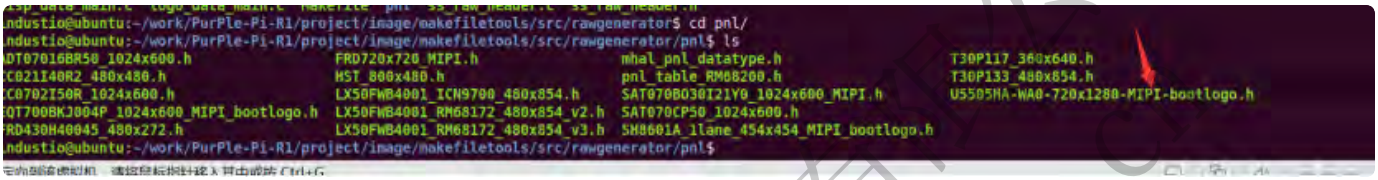
```
    80,  //MS_U16 m_wPanelOffTiming1;              ///<  time between back light & data while turn off power
    30,  //MS_U16 m_wPanelOffTiming2;              ///<  time between data & panel while turn off power

    ////////////////////////////////////////////
    // panel timing spec.
    ////////////////////////////////////////////
    // sync related
    HPW,  //MS_U8 m_ucPanelHSyncWidth;              ///<  VOP_01[7:0], PANEL_HSYNC_WIDTH
    HBP,  //MS_U8 m_ucPanelHSyncBackPorch;          ///<  PANEL_HSYNC_BACK_PORCH, no register setting, provide value for query only

                                                   ///<  not support Manuel VSync Start/End now
                                                   ///<  VOP_02[10:0] VSync start = Vtt - VBackPorch - VSyncWidth
                                                   ///<  VOP_03[10:0] VSync end = Vtt - VBackPorch
    VPW,  //MS_U8 m_ucPanelVSyncWidth;              ///<  define PANEL_VSYNC_WIDTH
    VBP,  //MS_U8 m_ucPanelVBackPorch;              ///<  define PANEL_VSYNC_BACK_PORCH

    // DE related
    (HPW+HBP),  //MS_U16 m_wPanelHStart;                  ///<  VOP_04[11:0], PANEL_HSTART, DE H Start (PANEL_HSYNC_WIDTH + PANEL_HSYNC_BACK_PORCH)
    (VPW+VBP),  //MS_U16 m_wPanelVStart;                  ///<  VOP_06[11:0], PANEL_VSTART, DE V Start
    HDA,  //MS_U16 m_wPanelWidth;                   ///< PANEL_WIDTH, DE width (VOP_05[11:0] = HEnd = HStart + Width - 1)
    VDA,  //MS_U16 m_wPanelHeight;                  ///< PANEL_HEIGHT, DE height (VOP_07[11:0], = Vend = VStart + Height - 1)

    // DClk related
    0,  //MS_U16 m_wPanelMaxHTotal;               ///<  PANEL_MAX_HTOTAL. Reserved for future using.
    (HDA+HPW+HBP+HFP),  //MS_U16 m_wPanelHTotal;                   ///< VOP_0C[11:0], PANEL_HTOTAL
    0,  //MS_U16 m_wPanelMinHTotal;               ///<  PANEL_MIN_HTOTAL. Reserved for future using.

    0,  //MS_U16 m_wPanelMaxVTotal;               ///<  PANEL_MAX_VTOTAL. Reserved for future using.
    (VDA+VPW+VBP+VFP),  //MS_U16 m_wPanelVTotal;                   ///< VOP_0D[11:0], PANEL_VTOTAL
```

```
    0,  //MS_U16 m_wPanelMinVTotal;                ///<  PANEL_MIN_VTOTAL. Reserved for future using.

    0,  //MS_U8 m_dwPanelMaxDCLK;                  ///<  PANEL_MAX_DCLK. Reserved for future using.
   ((unsigned long)(VDA+VPW+VBP+VFP)*(HDA+HPW+HBP+HFP)*FPS/1000000),  //MS_U8 m_dwPanelDCLK;                     ///<  LPLL_0F[23:0], PANEL_DCLK
        ,{0x3100_10[7:0], 0x3100_0F[15:0]}
    0,  //MS_U8 m_dwPanelMinDCLK;                  ///<  PANEL_MIN_DCLK. Reserved for future using.
                                                 ///<  spread spectrum
    0,  //MS_U16 m_wSpreadSpectrumStep;            ///<  move to board define, no use now.
    0,  //MS_U16 m_wSpreadSpectrumSpan;            ///<  move to board define, no use now.

    160,  //MS_U8 m_ucDimmingCtl;                    ///<  Initial Dimming Value
    255,  //MS_U8 m_ucMaxPWMVal;                     ///<  Max Dimming Value
    80,  //MS_U8 m_ucMinPWMVal;                     ///<  Min Dimming Value

    0,  //MS_U8 m_bPanelDeinterMode   :1;          ///<  define PANEL_DEINTER_MODE,  no use now
    E_MHAL_PNL_ASPECT_RATIO_WIDE,  //MHAL_DISP_PnlAspectRatio_e m_ucPanelAspectRatio; ///<  Panel Aspect Ratio, provide information to upper layer application for aspect ratio setting.
  /*
    *
    * Board related params
    *
    *  If a board ( like BD_MST064C_D01A_S ) swap LVDS TX polarity
    *    : This polarity swap value =
    *      (LVDS_PN_SWAP_H<<8) | LVDS_PN_SWAP_L from board define,
    *  Otherwise
    *    : The value shall set to 0.
    */
    0,  //MS_U16 m_u16LVDSTxSwapValue;
    E_MHAL_PNL_TI_8BIT_MODE,  //MHAL_DISP_ApiPnlTiBitMode_e m_ucTiBitMode;                        ///< MOD_4B[1:0], refer to "LVDS output app note"
    E_MHAL_PNL_OUTPUT_8BIT_MODE,  //MHAL_DISP_ApiPnlOutPutFormatBitMode_e m_ucOutputFormatBitMode;

    0,  //MS_U8 m_bPanelSwapOdd_RG    :1;          ///<  define PANEL_SWAP_ODD_RG
```

```
    0,  //MS_U8 m_bPanelSwapEven_RG   :1;          ///<  define PANEL_SWAP_EVEN_RG
    0,  //MS_U8 m_bPanelSwapOdd_GB    :1;          ///<  define PANEL_SWAP_ODD_GB
    0,  //MS_U8 m_bPanelSwapEven_GB   :1;          ///<  define PANEL_SWAP_EVEN_GB
```

如果感兴趣的可以了解一下PurPle-Pi-R1\project\image\makefiletools\src\rawgenerator\pnl\mhal_pnl_datatype.h和PurPle-Pi-R1\project\release\include\mi_panel_datatype.h这两个文件

1）.拷贝屏参文件到project/image/makefiletools/src/rawgenerator/pnl

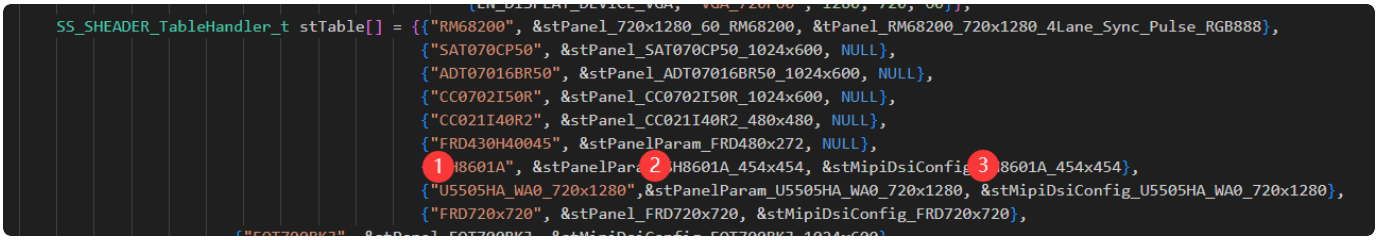


2）project/image/makefiletools/src/rawgenerator/disp_data_main.c，添加屏参的头文件以及修改配置和添加屏参文件：

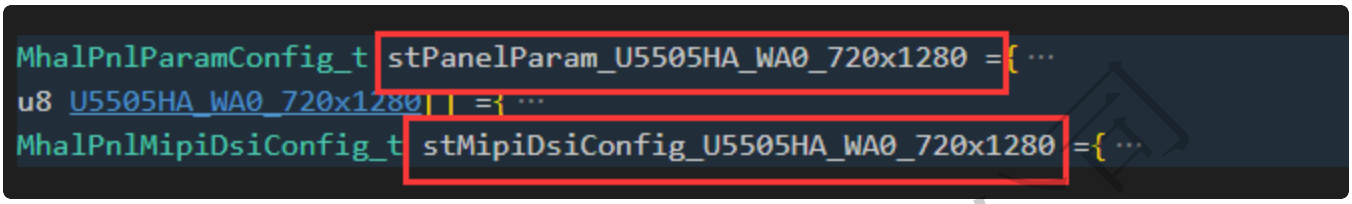
在修改配置的时候，我们可以看看定义的结构体变量中第一个是名称(随便取一般是屏幕型号)，第二个第三个对应屏参文件的机构提

Plain Text 复制代码

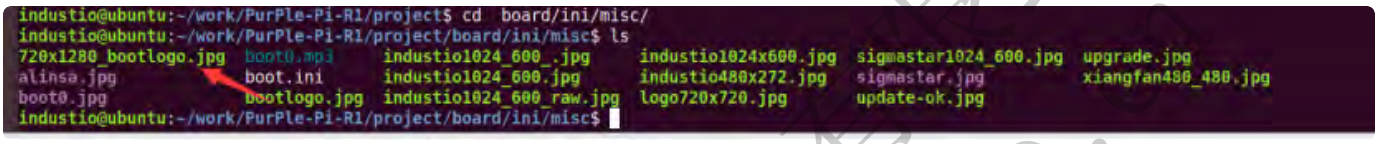
```
typedef struct
{
    const char *pName;
    MhalPnlParamConfig_t *pstMPnlParaConfig;
    MhalPnlMipiDsiConfig_t *pMipiDsiConfig;
}SS_SHEADER_TableHandler_t;
```



U5505HA-WA0-720x1280-MIPI-bootlogo.h



3） 在project/board/ini/misc添加屏参图片



4） 在project/configs/nvr/i2m/8.2.1/spinand.glibc.011a.128修改下配置信息

注意：这里的U5505HA_WA0_720x1280，要对应disp_data_main.c截图里的名称和图片的名称，如上图所示

5）配置完成之后编译一下：

Plain Text 复制代码

```
# cd PurPle-Pi-R1/project/image/makefiletools/src/rawgenerator

# make
```

6）如果发现开机启动的时候发现，图片没有起来，可能在uboot的时候没有使能背光和使能屏幕或者，你使能引脚是低给电的话，就需要在uboot设置下，所以我们可以在project/image/configs/i2m/script_nand.mk下添加：

```
        @echo setenv bootargs  $(rootfs$(BOOTENV)) $(kernel$(BOOTENV)) $(EXBOOTARGS) \$$\(mtdparts\) >> $(SCRIPTDIR)/set_config
ifeq ($(PROJECT), 2D06)
        @echo setenv bootcmd \' gpio output 4 1 \; gpio output 86 1 \;gpio output 14 1 \; bootlogo 0 0 0 0 0\; $(wifi24mclkcmd)\; $(wifirstoffcmd)\; nand read.e $(KERNELBOOTADDR) KERNEL $(kernel$(PATSIZE))\;$(rootfs$(BOOTCMD)) $(w
rstoncmd)\; bootm $(KERNELBOOTADDR)\;nand read.e $(KERNELBOOTADDR) RECOVERY $(kernel$(PATSIZE))\; bootm $(KERNELBOOTADDR) >> $(SCRIPTDIR)/set_config
else
        @echo setenv bootcmd \'gpio output 18 1\; bootlogo 0 0 0 0 0\; $(wifi24mclkcmd)\; $(wifirstoffcmd)\; nand read.e $(KERNELBOOTADDR) KERNEL $(kernel$(PATSIZE))\;$(rootfs$(BOOTCMD)) $(wifirstoncmd)\; bootm $(KERNELBOOTADDR)\;
d read.e $(KERNELBOOTADDR) RECOVERY $(kernel$(PATSIZE))\; bootm $(KERNELBOOTADDR) >> $(SCRIPTDIR)/set_config
```

7）然后重新一下固件

Plain Text　复制代码

```
# ./Release_to_customer.sh -f nand -p ssd202 -m 256
```

TP配置

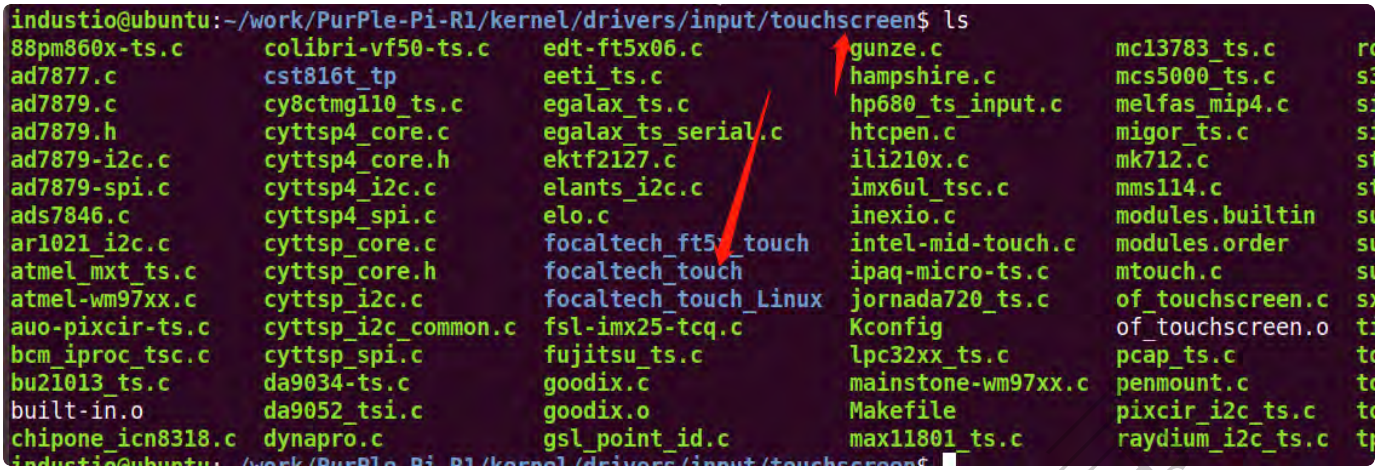
从TP的驱动IC的外观可以了解到，驱动IC型号为GT911：

很幸运，这并不是一个罕见的型号，内核已经自带驱动：

```
cainiaocl@dell-PowerEdge-R430:~/work/SSD20X/PurPle-Pi-R1/kernel/drivers/input/touchscreen$ ls
88pm860x-ts.c      cyttsp_core.h           hampshire.c          of_touchscreen.c   tsc2005.c
ad7877.c           cyttsp_i2c.c            hp680_ts_input.c     of_touchscreen.o   tsc2007.c
ad7879.c           cyttsp_i2c_common.c     htcpen.c             pcap_ts.c          tsc200x-core.c
ad7879.h           cyttsp_spi.c            ili210x.c            penmount.c         tsc200x-core.h
ad7879-i2c.c       da9034-ts.c             imx6ul_tsc.c         pixcir_i2c_ts.c    tsc40.c
ad7879-spi.c       da9052_tsi.c            inexio.c             raydium_i2c_ts.c   ts_icn85xx.c
ads7846.c          dynapro.c               intel-mid-touch.c    rohm_bu21023.c     ts_icn85xx.h
ar1021_i2c.c       edt-ft5x06.c            ipaq-micro-ts.c      s3c2410_ts.c       ucb1400_ts.c
atmel_mxt_ts.c     eeti_ts.c               jornada720_ts.c      silead.c           usbtouchscreen.c
atmel-wm97xx.c     egalax_ts.c             Kconfig              sis_i2c.c          w90p910_ts.c
auo-pixcir-ts.c    egalax_ts_serial.c      lpc32xx_ts.c         st1232.c           wacom_i2c.c
bcm_iproc_tsc.c    ektf2127.c              mainstone-wm97xx.c   stmpe-ts.c         wacom_w8001.c
bu21013_ts.c       elants_i2c.c            Makefile             sun4i-ts.c         wdt87xx_i2c.c
built-in.o         elo.c                   max11801_ts.c        sur40.c            wm831x-ts.c
chipone_icn8318.c  focaltech_ft5x_touch    mc13783_ts.c         surface3_spi.c     wm9705.c
colibri-vf50-ts.c  focaltech_touch         mcs5000_ts.c         sx8654.c           wm9712.c
cst816t_tp         focaltech_touch_Linux   melfas_mip4.c        ti_am335x_tsc.c    wm9713.c
cy8ctmg110_ts.c    fsl-imx25-tcq.c         migor_ts.c           touchit213.c       wm97xx-core.c
cyttsp4_core.c     fujitsu_ts.c            mk712.c              touchright.c       wrt_gslX680.c
cyttsp4_core.h     goodix.c                mms114.c             touchwin.c         wrt_gslX680.h
cyttsp4_i2c.c      goodix.o                modules.builtin      tps6507x-ts.c      zforce_ts.c
cyttsp4_spi.c      gsl_point_id.c          modules.order        ts4800-ts.c        zylonite-wm97xx.c
cyttsp_core.c      gunze.c                 mtouch.c             tsc2004.c
cainiaocl@dell-PowerEdge-R430:~/work/SSD20X/PurPle-Pi-R1/kernel/drivers/input/touchscreen$
```

配置kernel

把我们的屏幕去放置到：projecg/drivers/input/touchscreen下：



kernel需要配置加载GT911的驱动：

Plain Text 复制代码

```
# cd kernel
# ARCH=arm make menuconfig
```

Plain Text 复制代码

```
Location:
  -> Device Drivers  --->
     -> Input device support  --->
        -> [*] Touchscreens  --->
           ->  <*> Goodix I2C touchscreen
```

保存一下配置：

Plain Text 复制代码

```
 cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig
```

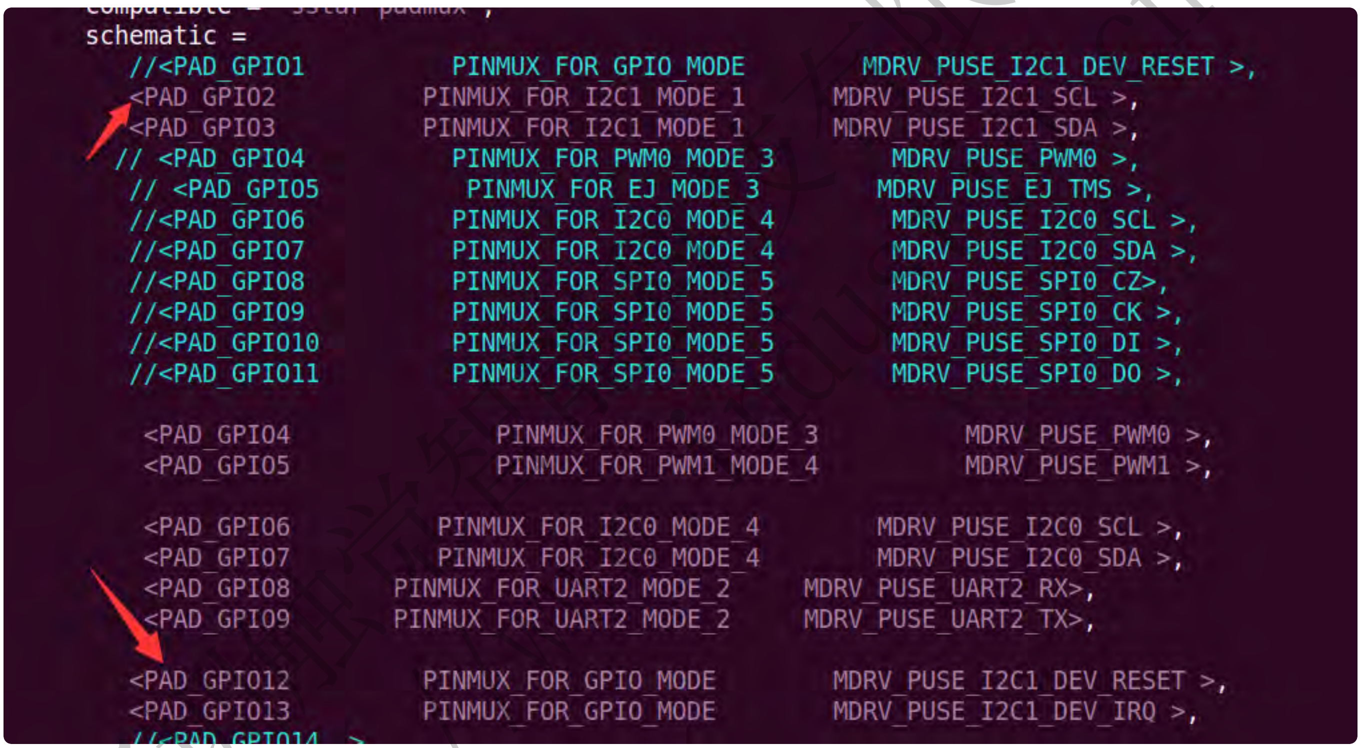
配置DTS

从原理图得知，I2C使用了GPIO1、GPIO2、GPIO12和GPIO13，根据SSD201 HW Checklist V6.xlsx，GPIO2和GPIO3该组I2C为I2C1，且MODE为1，因此I2C部分DTS作如下修改：

在设备书中配置I2C1，MODE 1模式：

Plain Text 复制代码

```
vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```

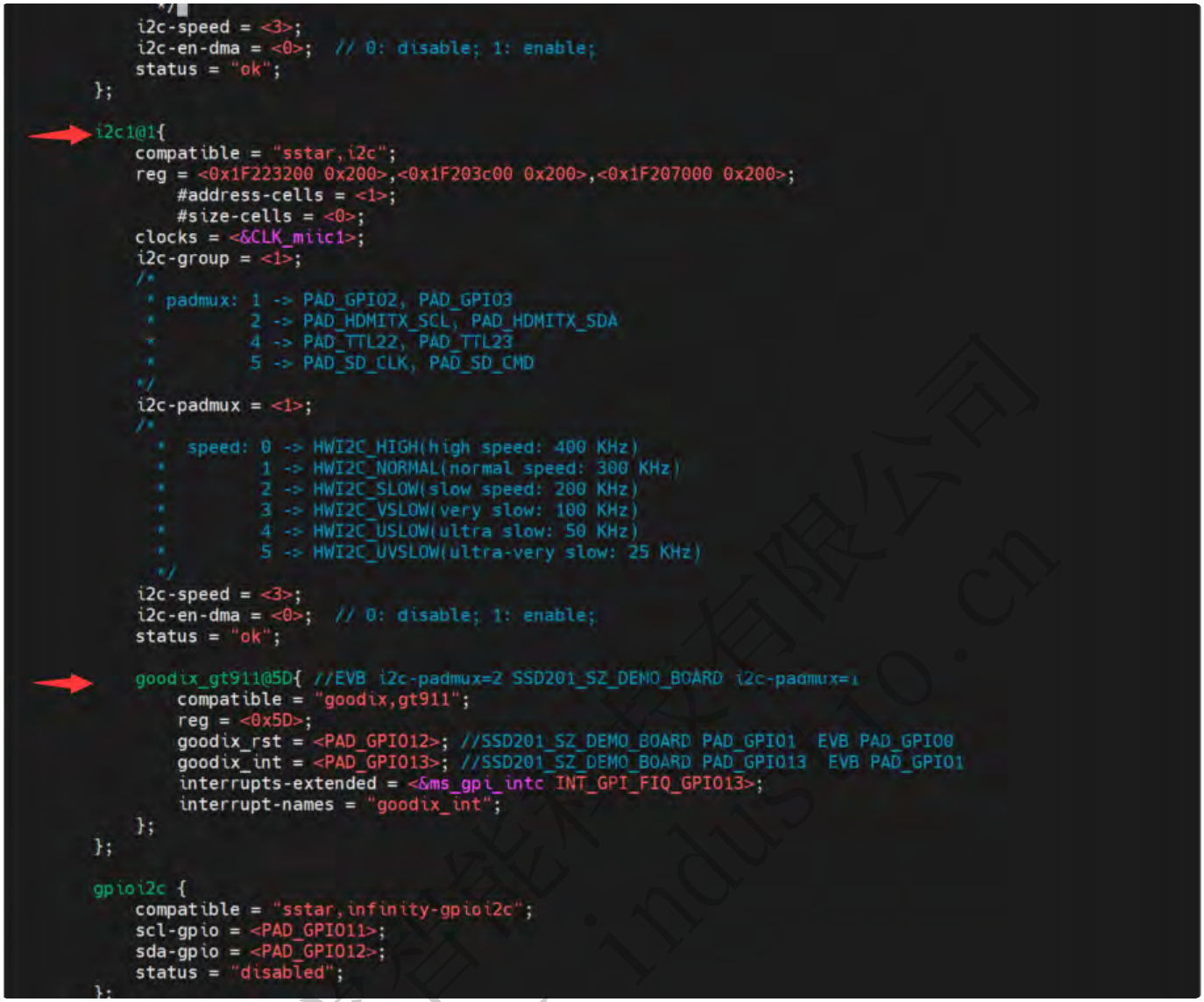


```
schematic =
    //<PAD_GPIO1          PINMUX_FOR_GPIO_MODE        MDRV_PUSE_I2C1_DEV_RESET >,
    <PAD_GPIO2          PINMUX_FOR_I2C1_MODE_1      MDRV_PUSE_I2C1_SCL >,
    <PAD_GPIO3          PINMUX_FOR_I2C1_MODE_1      MDRV_PUSE_I2C1_SDA >,
   // <PAD_GPIO4          PINMUX_FOR_PWM0_MODE_3        MDRV_PUSE_PWM0 >,
    // <PAD_GPIO5          PINMUX_FOR_EJ_MODE_3         MDRV_PUSE_EJ_TMS >,
    //<PAD_GPIO6          PINMUX_FOR_I2C0_MODE_4        MDRV_PUSE_I2C0_SCL >,
    //<PAD_GPIO7          PINMUX_FOR_I2C0_MODE_4        MDRV_PUSE_I2C0_SDA >,
    //<PAD_GPIO8          PINMUX_FOR_SPI0_MODE_5        MDRV_PUSE_SPI0_CZ>,
    //<PAD_GPIO9          PINMUX_FOR_SPI0_MODE_5        MDRV_PUSE_SPI0_CK >,
    //<PAD_GPIO10         PINMUX_FOR_SPI0_MODE_5        MDRV_PUSE_SPI0_DI >,
    //<PAD_GPIO11         PINMUX_FOR_SPI0_MODE_5        MDRV_PUSE_SPI0_DO >,

     <PAD_GPIO4             PINMUX_FOR_PWM0_MODE_3          MDRV_PUSE_PWM0 >,
     <PAD_GPIO5             PINMUX_FOR_PWM1_MODE_4          MDRV_PUSE_PWM1 >,

     <PAD_GPIO6           PINMUX_FOR_I2C0_MODE_4        MDRV_PUSE_I2C0_SCL >,
     <PAD_GPIO7           PINMUX_FOR_I2C0_MODE_4        MDRV_PUSE_I2C0_SDA >,
     <PAD_GPIO8         PINMUX_FOR_UART2_MODE_2     MDRV_PUSE_UART2_RX>,
     <PAD_GPIO9         PINMUX_FOR_UART2_MODE_2     MDRV_PUSE_UART2_TX>,

    <PAD_GPIO12         PINMUX_FOR_GPIO_MODE        MDRV_PUSE_I2C1_DEV_RESET >,
    <PAD_GPIO13         PINMUX_FOR_GPIO_MODE        MDRV_PUSE_I2C1_DEV_IRQ >,
    //<PAD_GPIO14 >
```

然后需要配置i2c1的节点信息：

Plain Text 复制代码

```
vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-rgb565-rmii-doublenet.dtsi
```



在i2c1下配置对应驱动的节点信息

Plain Text 复制代码

```
goodix_gt911@5D{ //EVB i2c-padmux=2 SSD201_SZ_DEMO_BOARD i2c-padmux=1
    compatible = "goodix,gt911";
    reg = <0x5D>;
    goodix_rst = <PAD_GPIO12>; //SSD201_SZ_DEMO_BOARD PAD_GPIO1  EVB PAD_GPIO0
    goodix_int = <PAD_GPIO13>; //SSD201_SZ_DEMO_BOARD PAD_GPIO13  EVB PAD_GPIO1
    interrupts-extended = <&ms_gpi_intc INT_GPI_FIQ_GPIO13>;
    interrupt-names = "goodix_int";
};


```

验证

配置完后，重新编译并更新kernel：

```
./Release_to_customer.sh -f nand -p ssd202 -m 256
```

系统起来后，可以看到/dev/input下有一个event0：

```
# ls /dev/input
```

并且在cat它的时候，点击TP，会输出信息：

```
# cat /dev/input/event0
```

测试：

我们使用测试程序：

```
# mkdir -p test/tp
# cd test/tp
# vi tp_test.c
```

C 复制代码

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <linux/input.h>

static int event0_fd = -1;
struct input_event ev0;

static int handle_event0()
{
    int rd;

    rd = read(event0_fd, &ev0, sizeof(struct input_event));
    if(rd < sizeof(struct input_event)){
        return 0;
    }

    if(EV_ABS == ev0.type){
        if (ev0.code == ABS_X){
            printf("ABS_X:");
        }else if (ev0.code == ABS_Y){
            printf("ABS_Y:");
        }else if (ev0.code == ABS_PRESSURE){
            printf("ABS_PRESSURE:");
        }else{
            printf("UNKNOWEN:");
        }
        printf("value:%d\n", ev0.value);
    }

    return 1;
}

int main(void)
{
    int done = 1;

    event0_fd = open("/dev/input/event0", O_RDONLY);
    if(event0_fd <0) {
        printf("open input device error\n");
        return -1;
    }
    while (done){
```

```
            done = handle_event0();
        }

        if(event0_fd > 0){
            close(event0_fd);
            event0_fd = -1;
        }
        return 0;
    }
```

交叉编译tp_test.c：

Plain Text 复制代码

```
# arm-linux-gnueabihf-gcc tp_test.c -o tp_test
```

将tp_test拷贝到板子上并运行：

Plain Text 复制代码

```
# ./tp_test
```

```
UNKNOWEN:value:367
UNKNOWEN:value:338
UNKNOWEN:value:366
UNKNOWEN:value:345
UNKNOWEN:value:365
UNKNOWEN:value:352
UNKNOWEN:value:364
UNKNOWEN:value:359
UNKNOWEN:value:363
UNKNOWEN:value:366
UNKNOWEN:value:362
UNKNOWEN:value:374
UNKNOWEN:value:361
UNKNOWEN:value:381
UNKNOWEN:value:360
UNKNOWEN:value:388
UNKNOWEN:value:359
UNKNOWEN:value:396
UNKNOWEN:value:358
UNKNOWEN:value:402
UNKNOWEN:value:357
UNKNOWEN:value:407
UNKNOWEN:value:356
UNKNOWEN:value:410
UNKNOWEN:value:355
UNKNOWEN:value:413
UNKNOWEN:value:354
UNKNOWEN:value:415
UNKNOWEN:value:417
UNKNOWEN:value:353
UNKNOWEN:value:418
UNKNOWEN:value:352
UNKNOWEN:value:420
UNKNOWEN:value:351
UNKNOWEN:value:422
UNKNOWEN:value:348
UNKNOWEN:value:424
UNKNOWEN:value:345
UNKNOWEN:value:426
UNKNOWEN:value:337
UNKNOWEN:value:328
[FTS_TS]fts_input_report_b:[B]P0 UP!
[FTS_TS]fts_input_report_b:[B]Points All Up!
UNKNOWEN:value:-1
```

GPIO配置

uboot 、kernel下使用GPIO

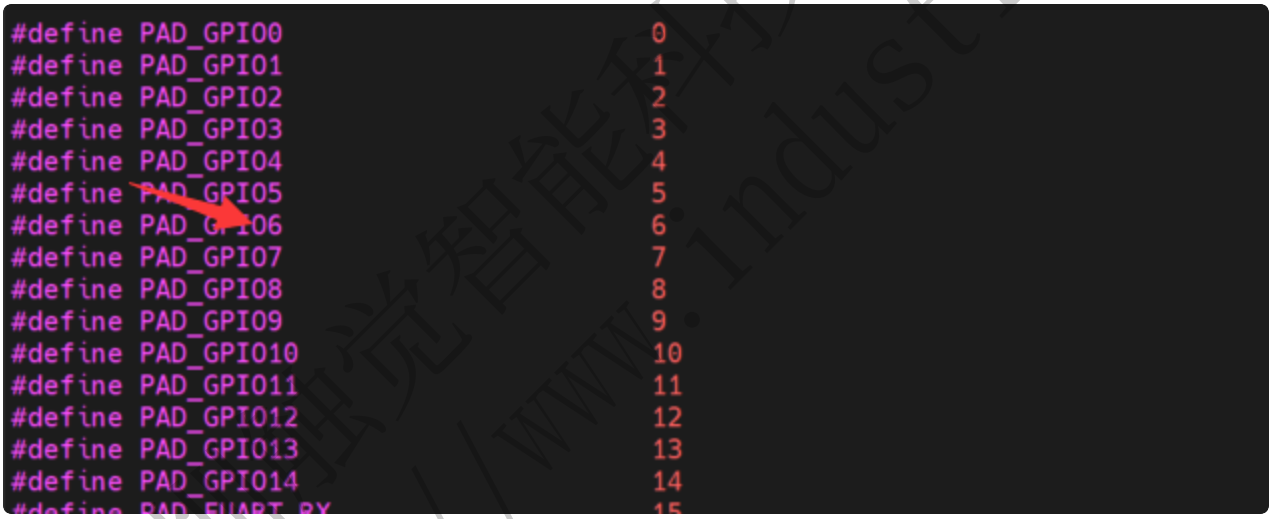
参考链接：https://wx.comake.online/doc/doc/SigmaStarDocs-SSD201-SIGMASTAR-202305231818/platform/BSP/Takoyaki/gpio_zh.html#425

user space 使用 GPIO

本次示例是使用GPIO6引脚如果不了解gpio的对应关系，我们可以直接查看gpio.h文件

Plain Text 复制代码

```
1  kernel/drivers/sstar/include/infinity2m/gpio.h
```

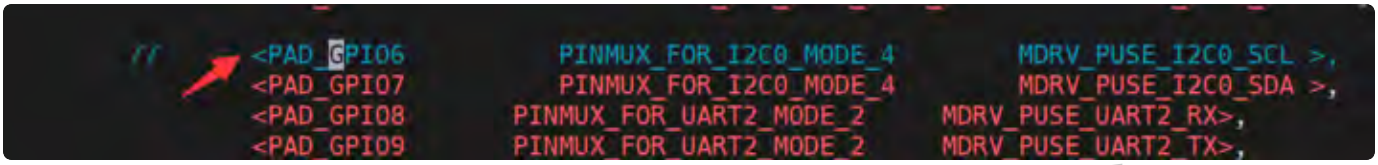


GPIO丝印名称	GPIO NUM
IO6	6

DTS的配置

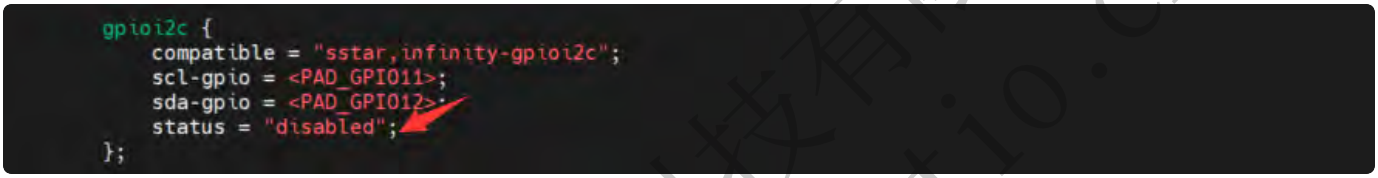
DTS需要将这GPIO设置为GPIO_MODE（取消其他复用，就是默认为GPIO_MODE）：

```
vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```




```
vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```


因为这个GPIO6可以复用为I2C，所以我们也需要把dts对应节点设置为disable：



kernel的配置

kernel本身是默认加载了GPIO的驱动，因此不需要修改任何东西：

每次修改需要保存一下配置：

```
# cp .config arch/arm/boot/dts/infinity2m-ssc011a-s01a-rgb565-rmii-doublenet.dtsi  -f
# cd ..
```

这样配置完成后，重新编译固件并升级kernel：

```
# cd ..
 # ./Release_to_customer.sh -f nand -p ssd202 -m 256
```

验证

我的验证方法是这样的，IO7（需另外配置）和IO6用导线联通。然后将IO7作为输入，IO6作为输出。如果IO7的输入电平等于IO6的输出电平，说明GPIO功能是正常的：

Plain Text 复制代码

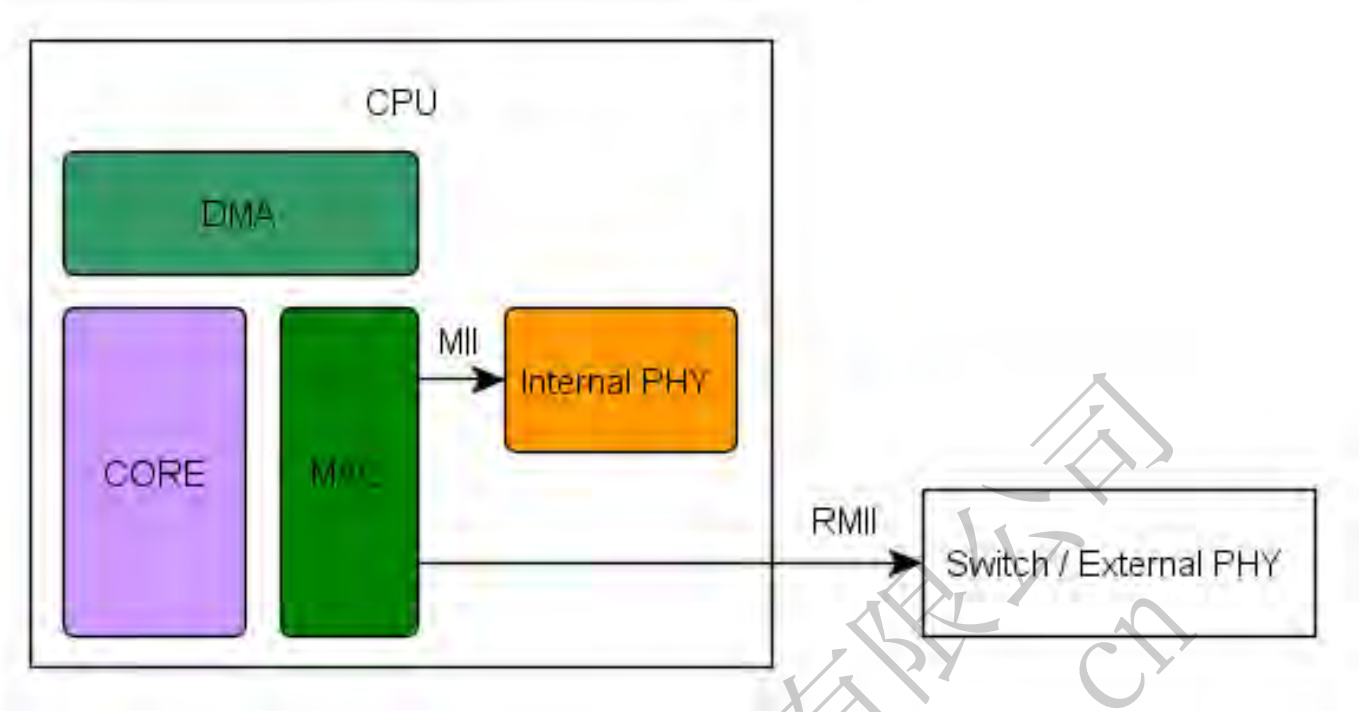
```
# echo 7 > /sys/class/gpio/export
# echo in > /sys/class/gpio/gpio7/direction
# echo 6 > /sys/class/gpio/export
# echo out > /sys/class/gpio/gpio6/direction
# echo 1 > /sys/class/gpio/gpio6/value
# cat /sys/class/gpio/gpio7/value
# echo 0 > /sys/class/gpio/gpio6/value
# cat /sys/class/gpio/gpio7/value
```

```
# echo 7 > /sys/class/gpio/export
[ss_gpi_intc_domain_alloc] hw:52 -> v:63
# echo in > /sys/class/gpio/gpio7/direction
# echo 6 > /sys/class/gpio/export
[ss_gpi_intc_domain_alloc] hw:51 -> v:64
# echo out > /sys/class/gpio/gpio6/direction
# echo 1 > /sys/class/gpio/gpio6/value
# cat /sys/class/gpio/gpio7/value
1
# echo 0 > /sys/class/gpio/gpio6/value
# cat /sys/class/gpio/gpio7/value
0
#
```

ETH配置

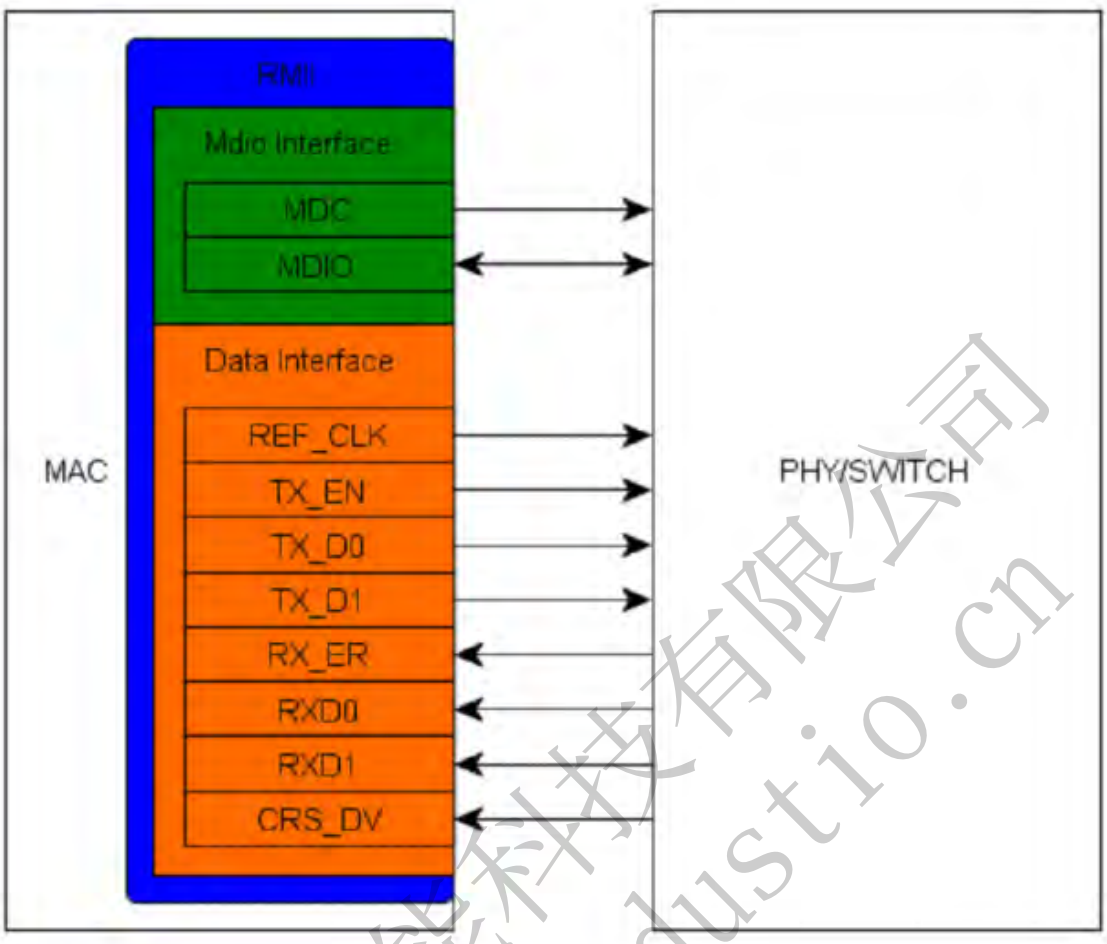
以太网框架图

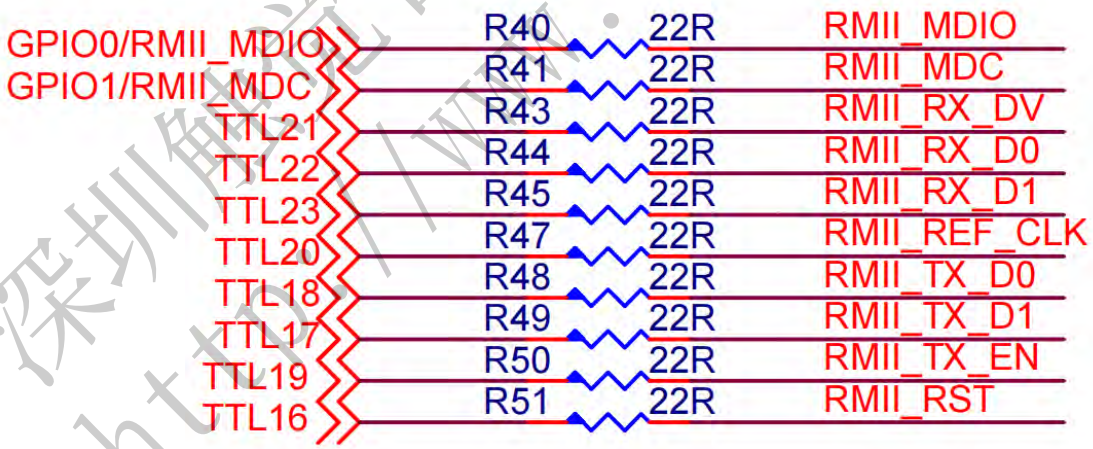
从硬件的角度看，以太网接口电路，主要由MAC（Media Access Control）控制器和物理层接口和PHY（Physical Layer，PHY）两大部分构成，如下图所示：



RMII接口介绍

我们从上图看出在MAC和PHY之间有一个RMII，这个是IEE-802.3定义的行业标准，是MAC与PHY之间的接口，这其中包括了数据接口和MDIO管理接口，其中MDIO管理接口包括了MDC和MDIO两根信号线，是用于PHY或者Switch芯片寄存器实现传输信息与状态控制。这其实是精简的MII接口，相对于MII节省了一般的数据线，相应的，他的是时钟也均采用50MHZ时钟源。





DTS的配置

MDIO和MDC对应的引脚为GPIO0和GPIO1，而且RMII其他数据线对应的是TTL16~23：故对应的引脚模式如下：

SSD201 Pin	PAD_Name	reg_ttl_mode 1					reg_ttl_mode 3				reg_ethl_mode 4	reg_ttl_mode 10	reg_ethl_mode 5
		28pin	RGB888				20pin	RGB565					
Pin56	PAD_TTL0	TTL_DOUT[0]	R0	G0	B0	R7/G7/B7	TTL_DOUT[0]	R3	R7	B3		TTL_DE	
Pin57	PAD_TTL1	TTL_DOUT[1]	R1	G1	B1	R6/G6/B6	TTL_DOUT[1]	R4	R6	B4		TTL_VSYNC	
Pin58	PAD_TTL2	TTL_DOUT[2]	R2	G2	B2	R5/G5/B5	TTL_DOUT[2]	R5	R5	B5		TTL_HSYNC	
Pin59	PAD_TTL3	TTL_DOUT[3]	R3	G3	B3	R4/G4/B4	TTL_DOUT[3]	R6	R4	B6		TTL_CK	
Pin60	PAD_TTL4	TTL_DOUT[4]	R4	G4	B4	R3/G3/B3	TTL_DOUT[4]	R7	R3	B7		TTL_DOUT[0]	
Pin61	PAD_TTL5	TTL_DOUT[5]	R5	G5	B5	R2/G2/B2	TTL_DOUT[5]	G2	G7	G2		TTL_DOUT[1]	
Pin65	PAD_TTL6	TTL_DOUT[6]	R6	G6	B6	R1/G1/B1	TTL_DOUT[6]	G3	G6	G3		TTL_DOUT[2]	
Pin66	PAD_TTL7	TTL_DOUT[7]	R7	G7	B7	R0/G0/B0	TTL_DOUT[7]	G4	G5	G4		TTL_DOUT[3]	
Pin67	PAD_TTL8	TTL_DOUT[8]	G0	B0	R0		TTL_DOUT[8]	G5	G4	G5		TTL_DOUT[4]	
Pin68	PAD_TTL9	TTL_DOUT[9]	G1	B1	R1		TTL_DOUT[9]	G6	G3	G6		TTL_DOUT[5]	
Pin69	PAD_TTL10	TTL_DOUT[10]	G2	B2	R2		TTL_DOUT[10]	G7	G2	G7		TTL_DOUT[6]	
Pin70	PAD_TTL11	TTL_DOUT[11]	G3	B3	R3		TTL_DOUT[11]	B3	B7	R3		TTL_DOUT[7]	
Pin71	PAD_TTL12	TTL_DOUT[12]	G4	B4	R4			B4	B6	R4		TTL_DOUT[8]	
Pin72	PAD_TTL13	TTL_DOUT[13]	G5	B5	R5		TTL_DOUT[13]	B5	B5	R5		TTL_DOUT[9]	
Pin73	PAD_TTL14	TTL_DOUT[14]	G6	B6	R6		TTL_DOUT[14]	B6	B4	R6		TTL_DOUT[10]	
Pin74	PAD_TTL15	TTL_DOUT[15]	G7	B7	R7		TTL_DOUT[15]	B7	B3	R7		TTL_DOUT[11]	
Pin79	PAD_TTL16	TTL_DOUT[16]	B0	R0	G0							TTL_DOUT[12]	
Pin80	PAD_TTL17	TTL_DOUT[17]	B1	R1	G1						ETH1_TXD1	TTL_DOUT[13]	
Pin81	PAD_TTL18	TTL_DOUT[18]	B2	R2	G2						ETH1_TXD0	TTL_DOUT[14]	
Pin82	PAD_TTL19	TTL_DOUT[19]	B3	R3	G3						ETH1_TX_EN	TTL_DOUT[15]	
Pin83	PAD_TTL20	TTL_DOUT[20]	B4	R4	G4						ETH1_TX_CLK		
Pin84	PAD_TTL21	TTL_DOUT[21]	B5	R5	G5						ETH1_COL		ETH1_TXD1
Pin85	PAD_TTL22	TTL_DOUT[22]	B6	R6	G6						ETH1_RXD0		ETH1_TXD0
Pin86	PAD_TTL23	TTL_DOUT[23]	B7	R7	G7						ETH1_RXD1		ETH1_TX_EN
Pin87	PAD_TTL24	TTL_CK					TTL_CK						ETH1_TX_CLK
Pin88	PAD_TTL25	TTL_HSYNC					TTL_HSYNC						ETH1_COL
Pin89	PAD_TTL26	TTL_VSYNC					TTL_VSYNC						ETH1_RXD0
Pin90	PAD_TTL27	TTL_DE					TTL_DE						ETH1_RXD1
Pin99	PAD_GPIO0										ETH1_MDIO		ETH1_MDIO
Pin100	PAD_GPIO1										ETH1_MDC		ETH1_MDC

故，我们在dts的配置如下：

Plain Text 复制代码

```
vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```

Plain Text 复制代码

```
<PAD_GPIO0          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,
<PAD_GPIO1          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,


<PAD_TTL16          PINMUX_FOR_GPIO_MODE        MDRV_PUSE_EMAC1_PHY_RESET
>,
<PAD_TTL17          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,
<PAD_TTL18          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,
<PAD_TTL19          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,
<PAD_TTL20          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,
<PAD_TTL21          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,
<PAD_TTL22          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,
<PAD_TTL23          PINMUX_FOR_ETH1_MODE_4          MDRV_PUSE_NA >,

```

设置设备树对应网络节点，如果只需要开启某一个网口，直接把status属性设置为disabled：

Plain Text 复制代码

```
# vi kernel/arch/arm/boot/dts/infinity2m-doublenet.dtsi
```

```
        emac0: emac0 {
            compatible = "sstar-emac";
            interrupts = <GIC_SPI INT_IRQ_EMAC IRQ_TYPE_LEVEL_HIGH>, <GIC_SPI INT_FIQ_LAN_ESD IRQ_TYPE_LEVEL_HIGH>;
            clocks = <&CLK_emac_ahb>,<&CLK_emac_tx>,<&CLK_emac_rx>;
            reg = <0x1F2A2000 0x800>, <0x1F343C00 0x600>, <0x1F006200 0x600>;
            pad = <0x1F203C38 0x0001 0x0000>; // pad selection from 0x0001
            phy-handle = <&phy0>;
            status = "ok";
            mdio-bus@emac0 {
                phy0: ethernet-phy@0 {
                    phy-mode = "mii";
                };
            };
        };

        emac1: emac1 {
            compatible = "sstar-emac";
            interrupts = <GIC_SPI INT_IRQ_EMAC_1 IRQ_TYPE_LEVEL_HIGH>, <GIC_SPI INT_FIQ_LAN_ESD IRQ_TYPE_LEVEL_HIGH>;
            clocks = <&CLK_emac_ahb>,<&CLK_emac1_tx>,<&CLK_emac1_rx>,<&CLK_emac1_tx_ref>,<&CLK_emac1_rx_ref>;
            reg = <0x1F2A2800 0x800>, <0x1F344200 0x600>, <0x00000000 0x000>;
            //pad = <0x1F203C38 0x0F00 0x0300>; // pad selection from 0x0100/0x0200/0x0300/0x0400/0x0500/0x0600/0x0700/0x0800/0x0900
            pad = <0x1F203C38 0x0F00 0x0300>; // pad selection from 0x0100/0x0200/0x0300/0x0400/0x0500/0x0600/0x0700/0x0800/0x0900
            status = "ok";
#if 1
            phy-handle = <&phy1>;
            mdio-bus@emac1 {//设置总线MDIO和PHY设备，下面使用rmii协议
                phy1: ethernet-phy@1 {
                    phy-mode = "rmii";
                };
            };
#else
            phy-mode = "rmii";
            fixed-link = <0 1 100 0 0>;
#endif
```

这里一般都是默认配好了的。

Kernel配置

在kernel目录下执行：

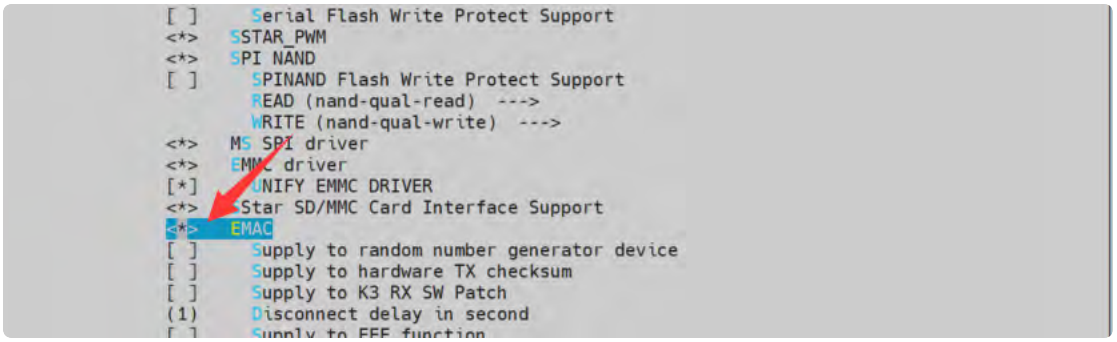
Plain Text 复制代码

```
make menuconfig ARCH=arm
```

在菜单里的：> Device Drivers > Network device support > PHY Device support and infrastructure 下选中MDIO bus/PHY：

```
        < >   Broadcom 7xxx SOCs internal PHYs
        < >   Broadcom BCM8706 and BCM8727 PHYs
        < >   Broadcom PHYs
        < >   Cicada PHYs
        < >   Davicom PHYs
        < >   Texas Instruments DP83848 PHY
        < >   Texas Instruments DP83867 Gigabit PHY
        -*-   MDIO Bus/PHY emulation with fixed speed/link PHYs
        < >   ICPlus PHYs
        < >   Intel XWAY PHYs
        < >   LSI ET1011C PHY
        < >   Intel LXT PHYs
        < >   Marvell PHYs
        < >   Micrel PHYs
```

然后在菜单里的：> Device Drivers > SStar SoC platform drivers下选中EMAC



即可，这些都是默认配置所以不需要自己配置。

保存下当前配置：

Plain Text 复制代码

```
# cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doubl
enet_defconfig -f
```

返回主目录重新编译：

Plain Text 复制代码

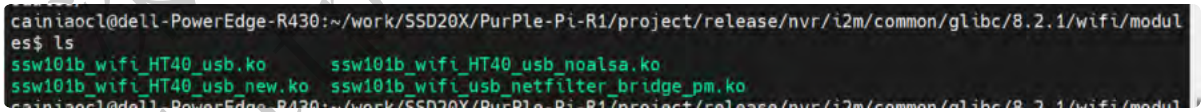
```
./Release_to_customer.sh -f nand -p ssd202 -m 256
```

WIFI配置

驱动配置

project配置

这里WIFI是指由原厂配套的WIFI，型号为ssw01b，SDK中包含了该WIFI的驱动模块，位于project/release/nvr/i2m/common/glibc/8.2.1/wifi/modules/ssw101b_wifi_HT40_usb.ko。



Plain Text 复制代码

```
# vi project/image/configs/i2m/rootfs.mk
```

```
if [ $(interface_wlan) = "enable" ]; then \
        mkdir -p  $(miservice$(RESOUCE))/wifi ; \
        if [ $(FLASH_TYPE) = "spinand" ]; then \
                cp -rf $(LIB_DIR_PATH)/wifi/libs/ap/*   $(miservice$(RESOUCE))/wifi ; \
                cp -rf $(LIB_DIR_PATH)/wifi/bin/ap/*   $(miservice$(RESOUCE))/wifi ; \
        fi;     \
        find $(LIB_DIR_PATH)/wifi/bin/ -maxdepth 1 -type f -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        find $(LIB_DIR_PATH)/wifi/bin/ -maxdepth 1 -type l -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        find $(LIB_DIR_PATH)/wifi/libs/ -maxdepth 1 -type f -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        find $(LIB_DIR_PATH)/wifi/libs/ -maxdepth 1 -type l -exec cp -P {}  $(miservice$(RESOUCE))/wifi \; ;\
        cp -rf $(LIB_DIR_PATH)/wifi/modules/*   $(miservice$(RESOUCE))/wifi ; \
        cp -rf $(LIB_DIR_PATH)/wifi/configs/*   $(miservice$(RESOUCE))/wifi ; \
fi;

```

我们可以看出我们需要把interface_wlan调成enable，这个的配置在：

```
# vi project/release/customer_tailor/nvr_i2m_display_glibc_tailor.mk
```

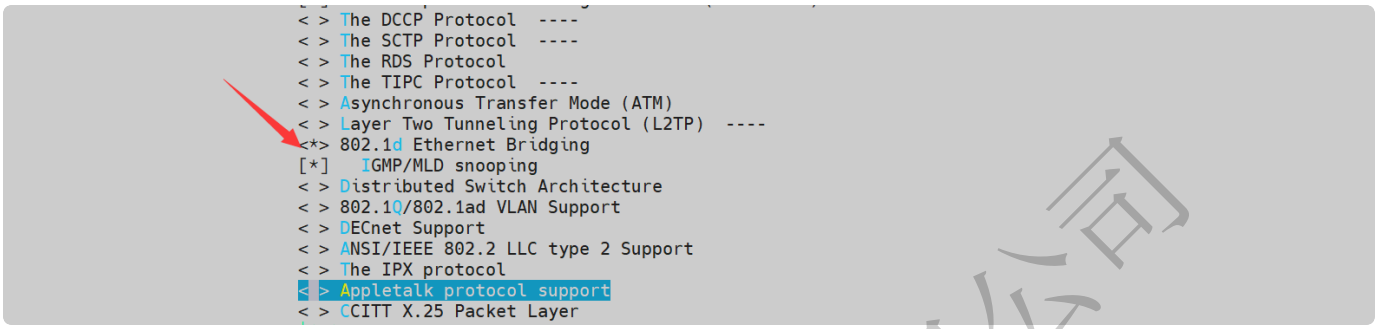
```
interface_rgn:=disable
interface_shadow:=disable
interface_uac:=disable
interface_vdf:=disable
interface_vdisp:=disable
interface_vdec:=enable
interface_venc:=enable
interface_wlan:=enable

misc_fbdev:=enable
#verify_zk_full_security:=enable
```

Kernel配置

进入kernel目录下进行配置：

```
ARCH=arm make menuconfig
```

> Networking support > Networking options。



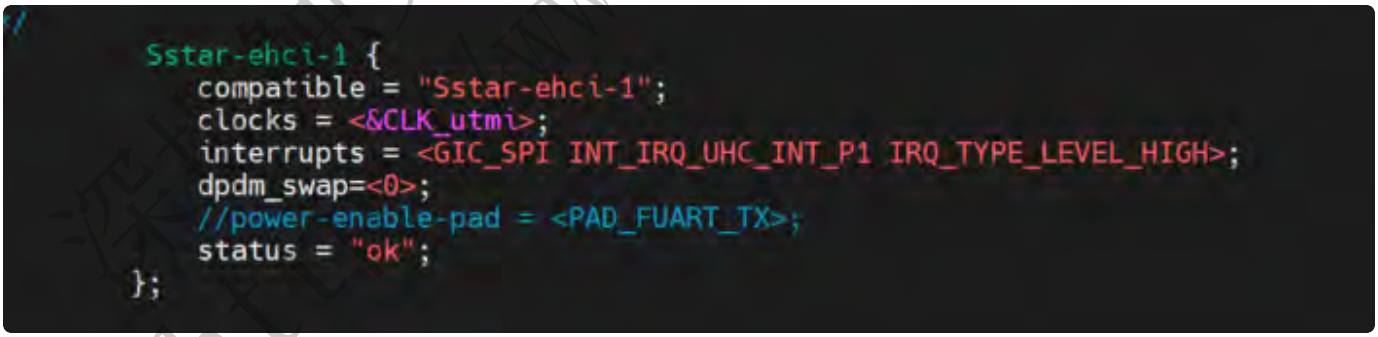
保存一下配置

```
cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig
```

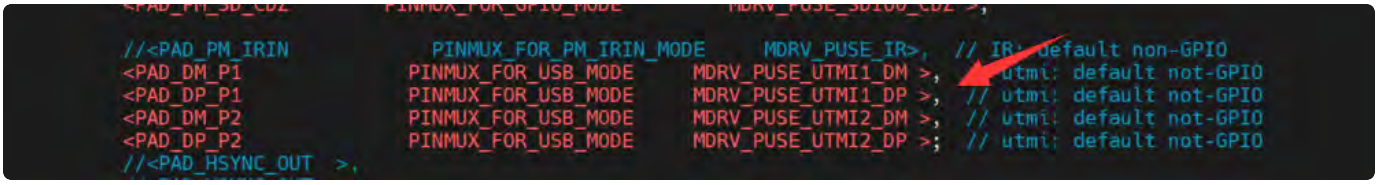
DTS的配置

根据原理图，我们的wifi使用的是usb1接口的，因此我们需要配置USB1（默认配好的，无需更改）：

```
vi kernel/arch/arm/boot/dts/infinity2m-doublenet.dtsi
```




```
vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi

```



系统启动后，通过lsusb可以看到1b20:8888的设备，它便是wifi模块。

加载驱动

执行我们的脚本文件自动加载驱动，在我们的脚本文件里已经包含所需要的驱动以及sta和AP所需要的一些套接字的路径：

Plain Text 复制代码

```
1  /config/wifi/ssw01bInit.sh
```

ssw01blnit.sh Plain Text 复制代码

```
#!/bin/sh

/config/riu_w e 30 11
/config/riu_w 103c 8 00
sleep 0.01
/config/riu_w 103c 8 10

#mkdir -p /etc/
#touch /etc/hosts
touch /appconfigs/hosts
mkdir -p /tmp/wifi/run
chmod 777 /tmp/wifi/run
mkdir -p /appconfigs/misc/wifi/
mkdir -p /var/wifi/misc/
mkdir -p /var/lib/misc/
mkdir -p /var/run/hostapd/
insmod /config/wifi/ssw101b_wifi_HT40_usb.ko
mdev -s
wlan0=`ifconfig -a | grep wlan0`
trial=0
maxtrycnt=50
while [ -z "$wlan0" ] && [ $trial -le $maxtrycnt ]
do
    sleep 0.2
    #echo currect try $trial...
    trial=$(($trial + 1 ))
    wlan0=`ifconfig -a | grep wlan0`
done
if [ $trial -le $maxtrycnt ]; then
    echo try $trial times
fi
if [ $trial -gt $maxtrycnt ];then
    echo wlan0 not found
    exit -1
fi
echo LOG_WARN=OFF > /sys/module/ssw101b_wifi_usb/Sstarfs/Sstar_printk_mask
echo LOG_INIT=OFF > /sys/module/ssw101b_wifi_usb/Sstarfs/Sstar_printk_mask
echo LOG_EXIT=OFF > /sys/module/ssw101b_wifi_usb/Sstarfs/Sstar_printk_mask
echo LOG_SCAN=OFF > /sys/module/ssw101b_wifi_usb/Sstarfs/Sstar_printk_mask
echo LOG_LMAC=OFF > /sys/module/ssw101b_wifi_usb/Sstarfs/Sstar_printk_mask
echo LOG_PM=OFF > /sys/module/ssw101b_wifi_usb/Sstarfs/Sstar_printk_mask
exit 0

```

```
# /config/wifi/ssw01bInit.sh
BANK:0xE 16bit-offset 0x30
0x0011
BANK:0x103C 16bit-offset 0x08
0x0001
BANK:0x103C 16bit-offset 0x08
0x0011
[Sstar_log]:Sstar_usb_module_init 0
[Sstar_log]:SVN_VER=1997,DPLL_CLOCK=2,BUILD_TIME=[===USB-ARES_B==
[Sstar_log]:Probe called -1 v1
[Sstar_log]:CONFIG_USE_DMA_ADDR_BUFFER TX_BUFFER_SIZE 800
[Sstar_log]:CONFIG_USB_AGGR_URB_TX enable cnt tx_dma_addr_buffer_end(  (null))tx_dma_addr_buffer(  (null)),0
[Sstar_log]:Sstar_usb_urb_malloc CONFIG_USE_DMA_ADDR_BUFFER max_num 4, total 131072
[Sstar_log]:CONFIG_USB_AGGR_URB_TX enable cnt tx_dma_addr_buffer_end(c6ac0000)tx_dma_addr_buffer(c6ae0000),8
[Sstar_log]:CONFIG_TX_NO_CONFIRM
```

驱动加载完后，我们就可以看到wlan0网卡：

```
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

p2p0      Link encap:Ethernet  HWaddr 16:C9:CF:50:71:19
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 14:C9:CF:50:71:19
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

STA模式

wpa_supplicant是一个独立运行的守护进程，用来启动无线网络后台服务，在消息循环中处理WPA状态机、控制命令、驱动事件、配置信息等。

常用命令参数如下：

Plain Text 复制代码

```
-I <ifname>          // 网络接口名称

-c <conf>            // 配置文件名称

-C <ctrl_intf>        // 控制接口名称

-D<driver>      // 驱动类型名称

-p <driver_param>    // 驱动参数

-b <br_ifname>       // 桥接口名称

-d              // 增加调试信息
```

前面我们已经加载了模块驱动，并且wlan0网卡存在，现在我们通过wpa_supplicant来连接WiFi热点，我们修改wpa_supplicant.conf，填入wifi热点信息：

```
vi /appconfigs/wpa_supplicant.conf
```


```
ctrl_interface=/tmp/wifi/run/wpa_supplicant
update_config=1
network={
        scan_ssid=1
        ssid="TP-LINK_B87A"
        psk="12345678"

}
```

我们可以去看看能否搜索到这个热点：

```
/config/wifi/iwlist wlan0 scan
```

接着尝试去连接：

```
/config/wifi/wpa_supplicant -D nl80211 -i wlan0 -c /appconfigs/wpa_supplica
nt.conf -B &
```

AP模式

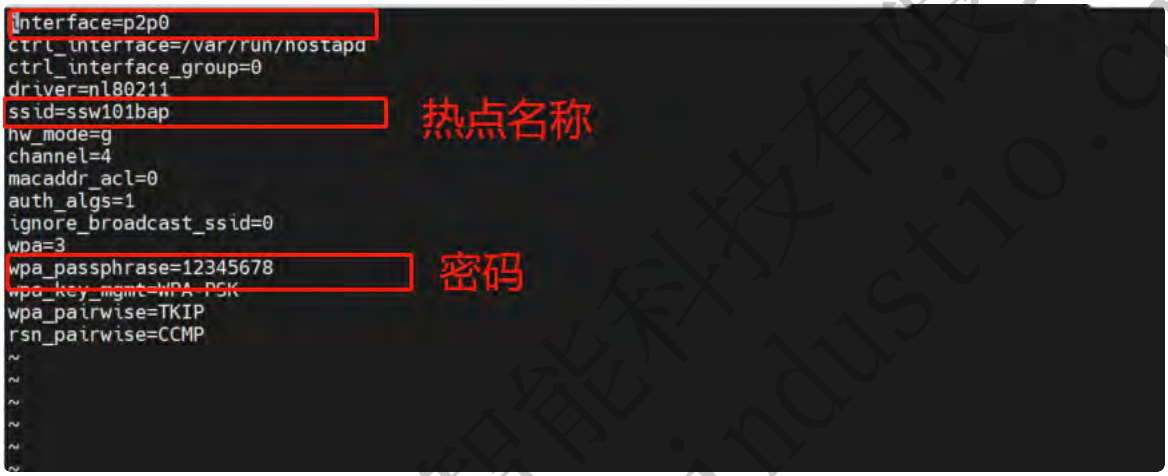
这里我们要以p2p0做为AP热点，且ip段为196.168.0.x，所以我们设置给ip地址，如果开启了服务ap服务及dhcp服务再设置ip的话，会出现连接失败的现象：

```
ifconfig p2p0 192.168.0.1
```

当我们的WiFi模块作为AP热点时，需要配置一下热点信息：

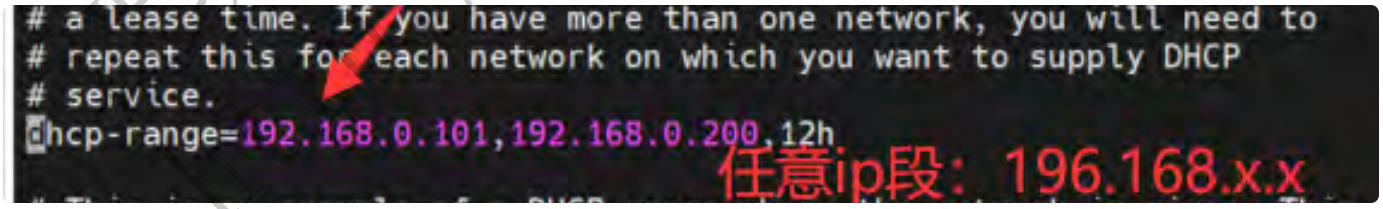
```
# vi /config/wifi/hostapd.conf
```

这里使用p2p0作为AP热点，当然我们也可以用wifi模块作为我们的AP热点，这样就要设置interface=wlan0，



开启DHCP服务，如果没有开启这项服务的话有可能会导致连接失败：

```
vi /config/wifi/dnsmasq.conf
```



```
# If you want dnsmasq to listen for DHCP and DNS requests only on
# specified interfaces (and the loopback) give the name of the
# interface (eg eth0) here.
# Repeat the line for more than one interface.
interface=p2p0
# Or you can specify which interface _not_ to listen on
#except-interface=
# Or which to listen on by address (remember to include 127.0.0.1 if
# you use this.)
#listen-address=
```

注意：这里的interface要根据那个模块作为AP热点来定，如果是wifi模块，则要设置为wlan0

开启热点

```
/config/wifi/hostapd -B /config/wifi/hostapd.conf
```

开启dhcp

```
# /config/wifi/dnsmasq -i wlan0 -C /config/wifi/dnsmasq.conf
```

```
p2p0      Link encap:Ethernet  HWaddr 16:C9:CF:50:71:19
          inet addr:192.168.0.1  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:988 (988.0 B)
```

上网

现在设备可以正常连接热点了，但此时我想连接设备能够上网，即把板子当作一个路由器，把eth0当作WAN，把wlan0当作LAN。首先需要确认eth0/eth1是可以上网的。

通过brctl桥接工具可以实现，此工具默认是没有安装的，和之前一样，从buildroot获取：

```
cd buildroot-2020.05/
ARCH=arm make menuconfig
```

```
Target packages  --->
    Networking applications  --->
        [*] bridge-utils
```

```
cp .config ./configs/ssd20x_defconfig -f
make BR2_JLEVEL=4
cd ../project/image/rootfs
rm rootfs/* -rf
cp ../../../buildroot-2020.05/output/images/rootfs.tar ./ -f
tar -xvf rootfs.tar -C ./rootfs/
tar -cvf rootfs.tar.gz ./rootfs
cd ../../../
```

重新编译固件

```
./Release_to_customer.sh -f nand -p ssd202 -m 256
```

更新固件后，前面的加载驱动、hostapd服务和dnsmasq服务需要重新执行，然后执行以下命令建立桥接：

```
brctl addbr br0
brctl addif br0 wlan0
brctl addif br0 eth0
ifconfig br0 up
```

RTC配置

kernel配置

RTC采用标准的LINUX框架，能够使用统一的接口来操作RTC。

所以我们进入内核菜单，开启一下RTC相关驱动：

```
make menuconfig ARCH=arm
```

Device Drivers

> SStar SoC platform drivers

```
< >   SW I2C via GPIO support
[*]   MS_RTC
< >     Mstar RTC driver
<*>     Mstar RTCWC driver
[ ]       Mstar RTCPWC driver Error Handle on Hw Exception
< >   watchdog driver
< >   sar driver
[*]   MMA HEAP
<*>   SSTAR 10/100 PHYs
```

保存一下配置并且重新编译：

Plain Text 复制代码

```
cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig
cd ..
./Release_to_customer.sh -f nand -p ssd202 -m 256
```

烧录到开发板，我们可以看到/dev/rtc节点，证明我们可以正常使用RTC。

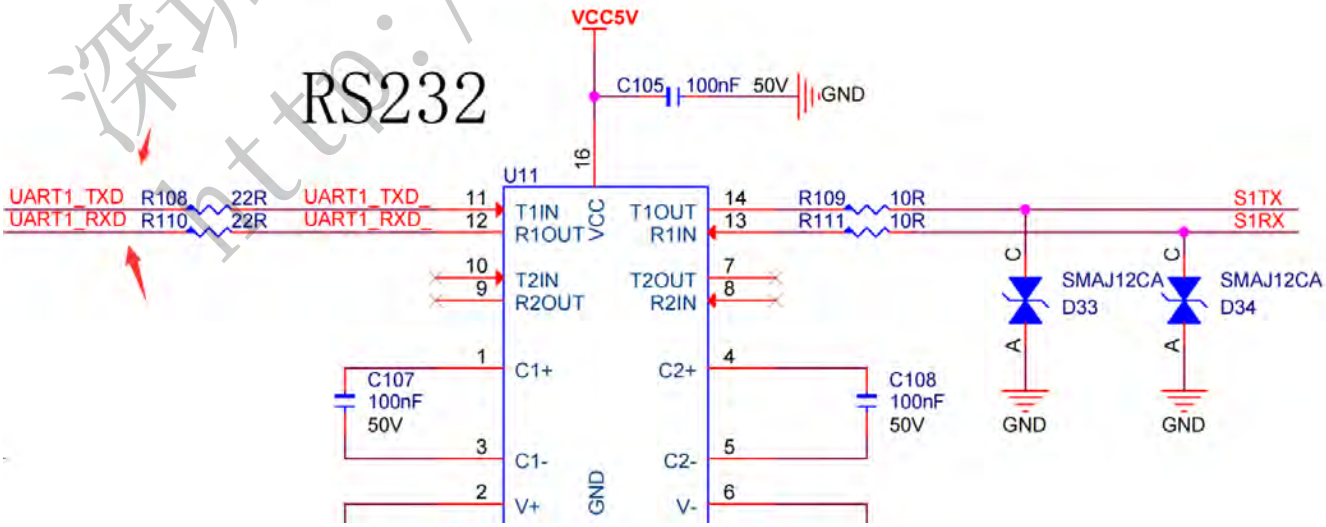
RTC操作方法

Plain Text 复制代码

```
# date -s "2020-11-12 10:10:10"   //设置系统时间
# hwclock -w                      //将系统时间更新到 RTC
# hwclock -r                       //读取 RTC 时间
# hwclock -s                      //将 RTC 时间同步系统时间
```

DTS配置

分析原理图，我们使用的是UART1_TXD和UART1_RXD引脚



Plain Text 复制代码

```
1  vi arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```

```
<PAD_TTL23          PINMUX_FOR_ETH1_MODE_4      MDRV_PUSE_NA >,
<PAD_UART1_RX       PINMUX_FOR_UART1_MODE_1     MDRV_PUSE_UART1_RX >,
<PAD_UART1_TX       PINMUX_FOR_UART1_MODE_1     MDRV_PUSE_UART1_TX >,
```

Plain Text 复制代码

```
1  vi arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```

音频（耳机）配置

1. LineOut

DTS配置

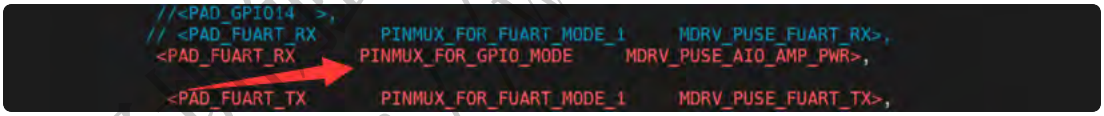
Bash 复制代码

```
1  vi kernel/arch/arm/boot/dts/infinity2m-doublenet.dtsi
```

```
sound {
    compatible = "sstar,audio";
//  reg = <0x1F000000 0x1000000>;
    interrupts=<GIC_SPI INT_IRQ_BACH IRQ_TYPE_LEVEL_HIGH>;
    playback-volume-level=<64>;    //0~94
    capture-volume-level=<64>;
   // micin-pregain-level=<1>;    //0~3
    micin-pregain-level=<0>;    //0~3
    micin-gain-level=<3>;    //0~7
    linein-gain-level=<2>;  //0~7
    amp-gpio = <PAD_FUART_RX 1>;//控制声道pin
    clocks = <&CLK_upll_384m>;
   // playback-dma-buffer=<98304>; //512(ms)*48(kHz)*2(ch)*2(16bits)
   // capture-dma-buffer=<122880>; //640(ms)*48(kHz)*2(ch)*2(16bits)
    digmic-padmux = <2>;
    i2s-padmux = <2>;//i2s mode
    keep-i2s-clk = <0>;
    status = "ok";
};

```

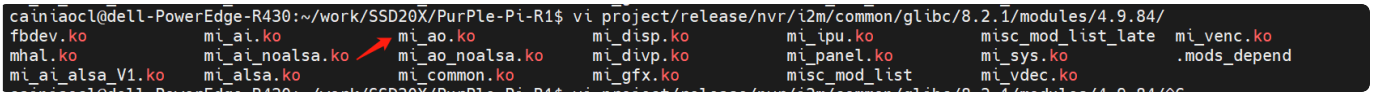
这里的amp-gpio，我们是需要开启的，因为在其他接口用到了PAD_FUART_RX，所以在使用LineOut之前是关闭的。这个的作用是为amp_gpio的左右声道控制pin：

```
vi kernel/arch/arm/boot/dts/infinity2m-ssc011a-s01a-padmux-rgb565-rmii-doublenet.dtsi
```



由于原厂没有提供驱动源码，只提供ko模块，模块在：

```
project/release/nvr/i2m/common/glibc/8.2.1/modules/4.9.84/
```

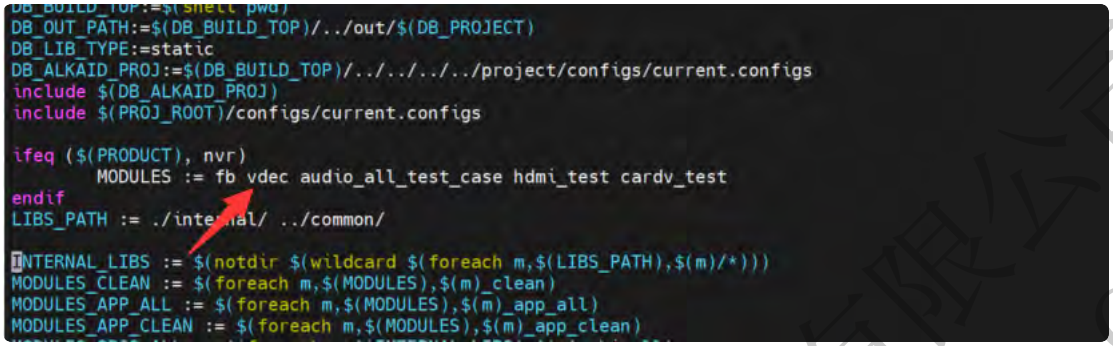


这个模块是linux内核驱动模块，它与音频相关，提供了音频设备交互，例如音频数据的输入，输出和处理，默认是直接编进内核的。

验证

我们音频播放程序audio_all_test_case验证lLineOut功能：

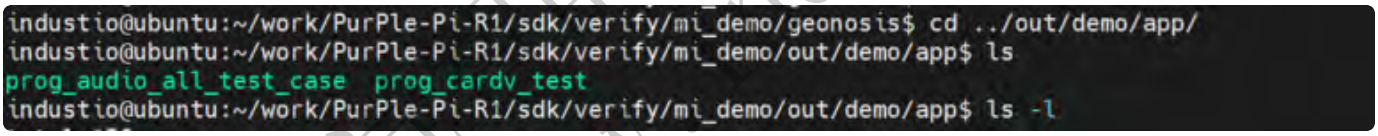
```Bash
cd sdk/verify/mi_demo/geonosis/
vi Makefile
```



我们可以看到，audio_all_test_cas默认是编译的，因此直接make即可：

```Bash
make
```

编译完成之后，将在../out/demo/app/下产生可执行程序：



我们把prog_audio_all_test_case放到project/image/rootfs_add_files/

```Bash
cp prog_audio_all_test_case ../../../../../../project/image/rootfs_add_files/usr/bin/
```

执行程序有了，缺少一个音频文件，这里注意，需要的是格式为wav的，我们把它放到

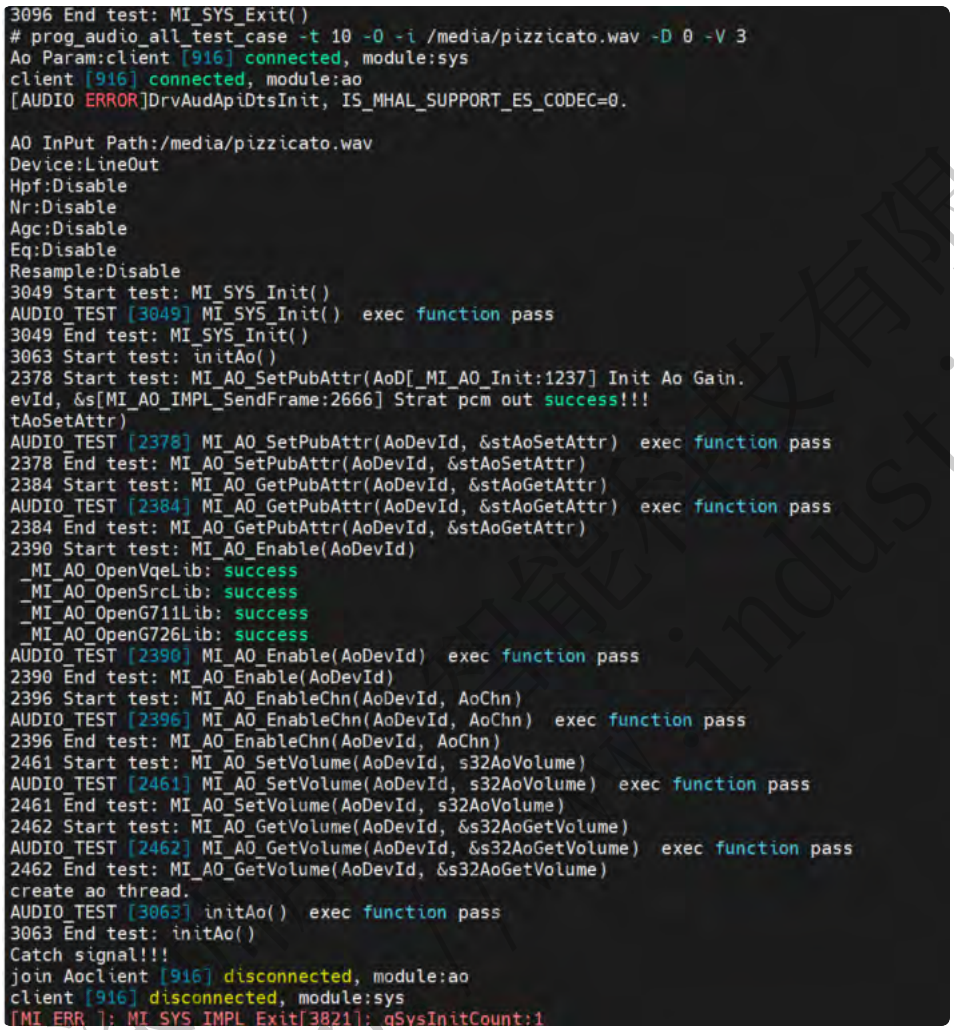
```Bash
mkdir project/image/rootfs_add_files/media
cp pizzicato.wav project/image/rootfs_add_files/media
```

重新编译系统

```
./Release_to_customer.sh -f nand -p ssd202 -m 256
```

在开发板内执行命令

```
prog_audio_all_test_case -t 10 -O -i /media/pizzicato.wav -D 0 -V 3
```



补充

1. LineOut支持左右声道，但是只有一路的dev，意思就是说可以连接到左右声道输出，但是只有一个音频通道可供使用，即只能输出单声道的音频播放，如果需要播放立体左右声道，需要音频文件本身就左右声道组合好通过STEREO MODE播放(不支持单独喂数据)

我们把测试demo和测试音频文件保存在IDO_SSD20X/开发板/IDO-SBC2D07/开发文档/test/lineout/下。

至此，LineOut调试完成。

2. DMIC

DTS的配置

DMIC接口也称双/立体声数字麦克风接口。 这种接口允许两个麦克风共享一个公共的时钟与数据线。每个麦克风被配置为在时钟信号的不同沿产生各自的输出。这样两个麦克风的输出就能保持相互同步，设计师就能确保来自每个通道的数据被同时捕获到。

首先根据我们的硬件确定使用的是那组的pin，使用SSD201HWChecklistV6.xlsx 可以看出是：

Pin Location	Power Domain default 3.3V	Ball Pin Name	GPIO index	default value after reset	default Rpu/Rpu	5V tolerance	Driving	Interrupt Available edge, rising/falling	mode	reg_pwm3_mode	reg_dmic_mode
										3	2
										1pin	3pin
5	VDDP_0	PAD_GPIO85	85	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes			
6	VDDP_0	PAD_GPIO86	86	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes			
7	VDDP_0	DMIC_R	87	input, pull-up	Rpu, 90K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes			DMIC_D1
8	VDDP_0	DMIC_L	88	input, pull-up	Rpu, 90K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes			DMIC_D0
9	VDDP_0	DMIC_CLK	89	input, pull-down	Rpd, 103K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes			DMIC_CLK

Plain Text 复制代码

```
sound {
    compatible = "sstar,audio";
//  reg = <0x1F000000 0x1000000>;
    interrupts=<GIC_SPI INT_IRQ_BACH IRQ_TYPE_LEVEL_HIGH>;
    playback-volume-level=<64>;    //0~94
    capture-volume-level=<64>;
   // micin-pregain-level=<1>;    //0~3
    micin-pregain-level=<0>;    //0~3
    micin-gain-level=<3>;    //0~7
    linein-gain-level=<2>;  //0~7
    amp-gpio = <PAD_FUART_RX 1>;//控制声道pin
    clocks = <&CLK_upll_384m>;
   // playback-dma-buffer=<98304>; //512(ms)*48(kHz)*2(ch)*2(16bits)
   // capture-dma-buffer=<122880>; //640(ms)*48(kHz)*2(ch)*2(16bits)
    digmic-padmux = <2>;
    i2s-padmux = <2>;//DMIC mode
    keep-i2s-clk = <0>;
    status = "ok";
};

```

配置是跟上面LineOut的没啥关系的，这里主要注意的是i2s-padmux，这里对应的是mode 2，上面也有说明。

我们修改对应的padmux dts，设置引脚为DMIC_MODE：

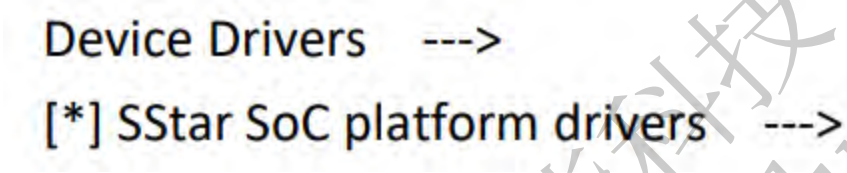
```
        //<PAD_HSYNC_OUT  >,
        //<PAD_VSYNC_OUT  >,
        //<PAD_HDMITX_SCL         PINMUX_FOR_I2C0_MODE_1      MDRV_PUSE_I2C0_SCL >,
        //<PAD_HDMITX_SDA         PINMUX_FOR_I2C0_MODE_1      MDRV_PUSE_I2C0_SDA >,
        <PAD_HDMITX_SCL PINMUX_FOR_DMIC_MODE_2 MDRV_PUSE_DMIC_D1 >,
        <PAD_HDMITX_SDA PINMUX_FOR_DMIC_MODE_2 MDRV_PUSE_DMIC_D0 >,
        <PAD_HDMITX_HPD PINMUX_FOR_DMIC_MODE_2 MDRV_PUSE_DMIC_CLK >;

        //<PAD_HDMITX_HPD  >,
        //<PAD_HSYNC_OUT          PINMUX_FOR_IDAC_MODE        MDRV_PUSE_IDAC_HSYNC>,
        //<PAD_VSYNC_OUT          PINMUX_FOR_IDAC_MODE        MDRV_PUSE_IDAC_VSYNC>;
    status = "ok"; // ok or disable
                                                                    128,16        92%
```

kernel

当然，kernel 也需要加载对应的驱动：

Plain Text 复制代码

```
# cd kernel
# ARCH=arm make menuconfig
```



保存一下配置：

Plain Text 复制代码

```
cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig
```

接下来，我们编译下固件，然后烧进开发板，测试一下：

Plain Text 复制代码

```
prog_audio_all_test_case -t 20 -I -o /tmp -d 1 -m 0 -c 2 -s 48000
```

这样将在/tmp产生我们的录音文件：

3. AMIC

AMIC功能是默认加载，不需要修改 DTS 配置。和 DMIC 一样，使用 audio_all_test_case 程序来测试：

```
prog_audio_all_test_case -t 20 -I -o /tmp -d 0 -m 0 -c 2 -s 48000
```

这里跟DMIC唯一不同的是，-d 0，在DIMC中，是-d 1

```
----- audio all test -----
-t : AI/AO run time(s)
AI Device Id: Amic[0] Dmic[1] I2S RX[2] Linein[3] Amic+I2S RX[4] Dmic+I2S RX[5]
AI test option :
-I : Enable AI
-o : AI output Path
-d : AI Device Id
-m : AI Sound Mode: Mono[0] Stereo[1] Queue[2]
-c : AI channel count
-s : AI Sample Rate
-v : AI Volume
-h : AI Enable Hpf
-g : AI Enable Agc
-e : AI Enable Eq
-n : AI Enable Nr
-r : AI Resample Rate
-a : AI Aenc Type [g711a g711u g726_16 g726_24 g726_32 g726_40]
-A : AI Enable Aed
-b : AI Enable SW Aec
-S : AI Enable Ssl
-F : AI Enable Beamforming
-M : AI mic disttance for Ssl/Bf (cm, step 1 cm)
-C : AI Bf doc (0~180)
-w : AI Aec Working Sample Rate(Not necessary)
-W : AI enable dump pcm data
```

将会在/tmp/目录下生成 Chn0_Amic_48K_16bit_MONO.wav 和 Chn1_Amic_48K_16bit_MONO.wav。如果有LineOut接口，可以直接播放该录音文件，判断是否录音成功。

4. I2S

DTS的配置

首先根据我们的硬件确定使用的是那组的pin，使用SSD201HWChecklistV6.xlsx 可以看出是：

Pin Location	Power Domain default 3.3V	Ball Pin Name	GPIO index	default value after reset	default Rpu/Rpu	5V tolerance	Driving	Interrupt Available edge, rising/falling	reg_i2c1_mode	reg_i2s_mode	reg_i2s_mode
									5	1	3
									2pin	4pin	4pin
94	VDDP_1	PAD_SD_CLK	51	input, pull-down	Rpd, 103K ohm	Yes	DRV=1 ->8mA DRV=0 ->4mA DRV=1 ->8mA	Yes	I2C1_SCL		I2S_BCK
95	VDDP_1	PAD_SD_CMD	52	input, pull-up	Rpu, 90K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes	I2C1_SDA		I2S_SDI
96	VDDP_1	PAD_SD_D3	56	input, pull-up	Rpu, 90K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes			I2S_SDO
97	VDDP_1	PAD_SD_D2	55	input, pull-up	Rpu, 90K ohm	Yes	DRV=0 ->4mA DRV=1 ->8mA	Yes			
99	VDDP_1	PAD_GPIO0	0	input, pull-down	Rpd, 103K ohm	Yes	>4mA	Yes		I2S_WCK	
100	VDDP_1	PAD_GPIO1	1	input, pull-down	Rpd, 103K ohm	Yes	>4mA	Yes		I2S_BCK	
101	VDDP_1	PAD_GPIO2 (I2C1_SCL)	2	input, pull-down	Rpd, 103K ohm	Yes	>4mA	Yes		I2S_SDI	
102	VDDP_1	PAD_GPIO3 (I2C1_SDA)	3	input, pull-down	Rpd, 103K ohm	Yes	>4mA	Yes		I2S_SDO	

然后修改dts的sound节点i2s-padmux 的值（mode的值）：

```
sound {
    compatible = "sstar,audio";
//  reg = <0x1F000000 0x1000000>;
    interrupts=<GIC_SPI INT_IRQ_BACH IRQ_TYPE_LEVEL_HIGH>;
    playback-volume-level=<64>;    //0~94
    capture-volume-level=<64>;
   // micin-pregain-level=<1>;    //0~3
    micin-pregain-level=<0>;    //0~3
    micin-gain-level=<3>;    //0~7
    linein-gain-level=<2>;  //0~7
    amp-gpio = <PAD_FUART_RX 1>;//控制声道pin
    clocks = <&CLK_upll_384m>;
   // playback-dma-buffer=<98304>; //512(ms)*48(kHz)*2(ch)*2(16bits)
   // capture-dma-buffer=<122880>; //640(ms)*48(kHz)*2(ch)*2(16bits)
    digmic-padmux = <2>;
    i2s-padmux = <1>;//i2s mode
    keep-i2s-clk = <0>;
    status = "ok";
};

```

接着还需要将该组 GPIO 配置为 I2S 模式：

```
<PAD_GPIO0          PINMUX_FOR_I2s_MODE_1      MDRV_PUSE_I2s_WCK >,
<PAD_GPIO1          PINMUX_FOR_I2s_MODE_1      MDRV_PUSE_I2s_BCK >,
<PAD_GPIO2          PINMUX_FOR_I2s_MODE_1      MDRV_PUSE_I2s_SDI >,
<PAD_GPIO3          PINMUX_FOR_I2s_MODE_1      MDRV_PUSE_I2s_SDO >,
```

这样配置后，和 AMIC/DMIC 一样，使用 audio_all_test_case 程序来测试：

```
prog_audio_all_test_case -t 20 -I -o /tmp -d 2 -m 0 -c 2 -s 48
```

将会在/tmp/目录下生成Chn0_I2SRx_48K_16bit_MONO.wav和Chn1_I2SRx_48K_16bit_MONO.wav。如果有LineOut接口，可以直接播放该录音文件，判断是否录音成功。

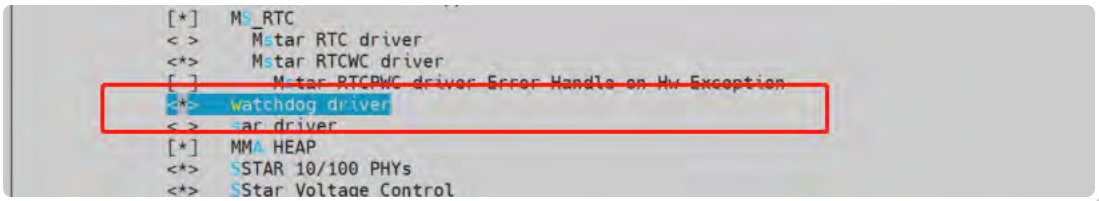
Watchdog配置

开启驱动

> Device Drivers

> [*]SStar SoC platform drivers

> <*>watchdog driver



保存下配置

```
cp .config arch/arm/configs/infinity2m_spinand_ssc011a_s01a_minigui_doublenet_defconfig
```

编译固件烧录开发板

如果能查找到节点/dev/watchdog就说明加载成功

测试

使用标准的watchdog命令即可对watchdog进行测试。

```
# watchdog -t 10 /dev/watchdog     //表示每10s喂狗一次
# watchdog -T 5  /dev/watchdog     //表示超过5s没有喂狗则系统重启
```

以下测试没有在规定时间内喂狗，系统重启功能：

```
# watchdog -t 10 -T 5 /dev/watchdog  //5s后系统重启
```

以下测试在规定时间内喂狗，系统正常运行：

```
# watchdog -t 10 -T 60 /dev/watchdog
```

添加 watchdog 服务

Shell 复制代码

```shell
#!/bin/sh
#Start watchdog
case"$1"in
    start)
        echo "Startingwatchdog..."
        watchdog -t 3 /dev/watchdog
        ;;
    stop)
        ;;
    restart|reload)
        ;;
    *)
        echo "Usage:$0{start|stop|restart}"
        exit 1
    esac
        exit $?
```

这个服务设定每3秒喂狗一次，而默认是60秒没有喂狗系统才重启，因此系统正常运行情况下，看门狗不会复位。