君正®

Halley5 Linux4.4 内核开发手册

Author: 系统软件部

Version: 2.0
Date: 2020.9.7

北京君正集成电路股份有限公司
Ingenic Semiconductor Co., Ltd.

君正®

Halley5 Linux4.4 内核开发手册

Release history

Date	Revision	Change
2020.12.10	1.0	First release
2021.9.7	2.0	1. add sadc 模块 2. 更新 efuse，dpu 模块。 3. 修改 ISP 模块中使用 ffmpeg 预览的方法

Disclaimer

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

Ingenic Semiconductor Co., Ltd.

Ingenic Headquarters, East Bldg. 14, Courtyard #10
Xibeiwang East Road, Haidian District, Beijing, China,
Tel: 86-10-56345000
Fax:86-10-56345001
Http: //www.ingenic.com

北京君正集成电路股份有限公司

地址:北京市海淀区东北西路中关村软件园二期君正总部大楼
电话: 86-10-56345000
传真: 86-10-56345001
http: //www.ingenic.com

目　录

关 键 词	文档 格式
摘　　要	本文档说明文档的基本格式。
缩 略 语	
参考资料	

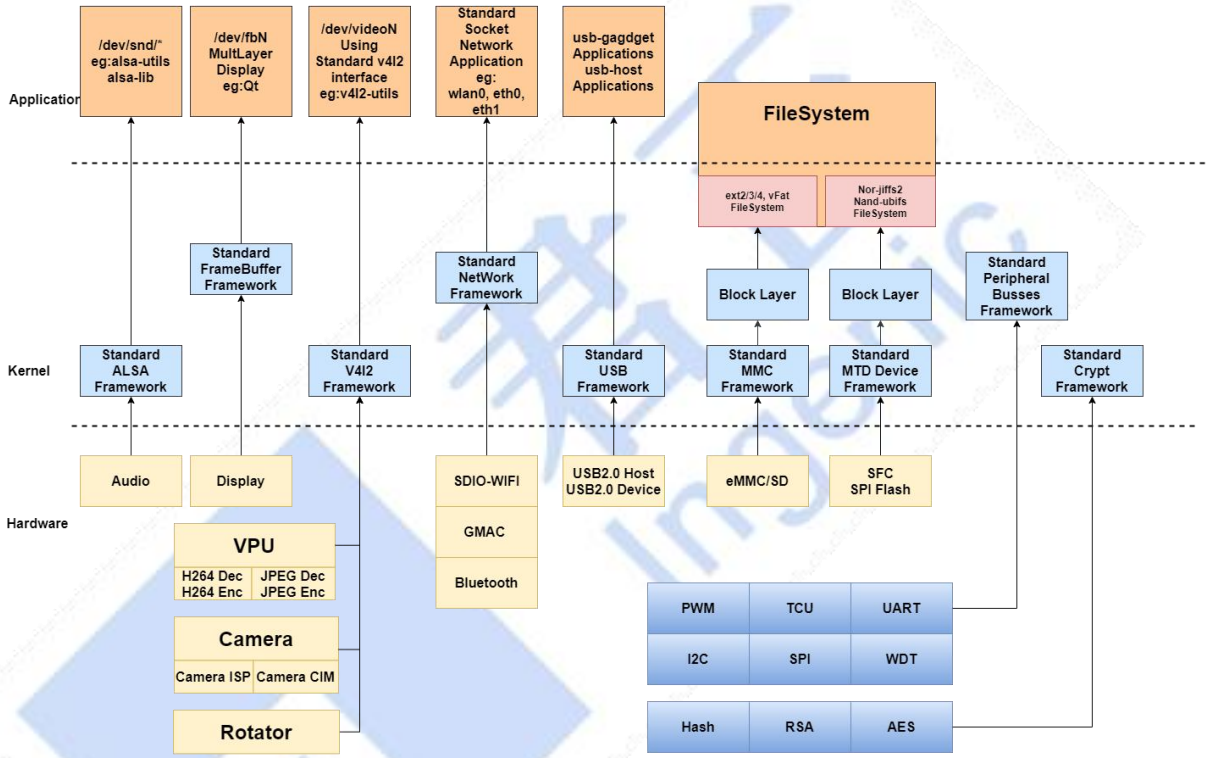
1 内核开发简介

该文档针对君正系列芯片，Kernel4.4 标准内核平台进行研发，主要针对内核的基本结构，内核的编译，设备树和驱动的配置和裁剪进行说明。

1.1 内核基本结构

君正系列芯片的控制器在 Kernel4.4 进行了驱动层的适配，所使用的驱动接口也都是用了内核的标准驱动接口，用户可以基于标准应用接口进行二次开发，而不用关心底层的具体实现。

内核基本适配结构图如下图:



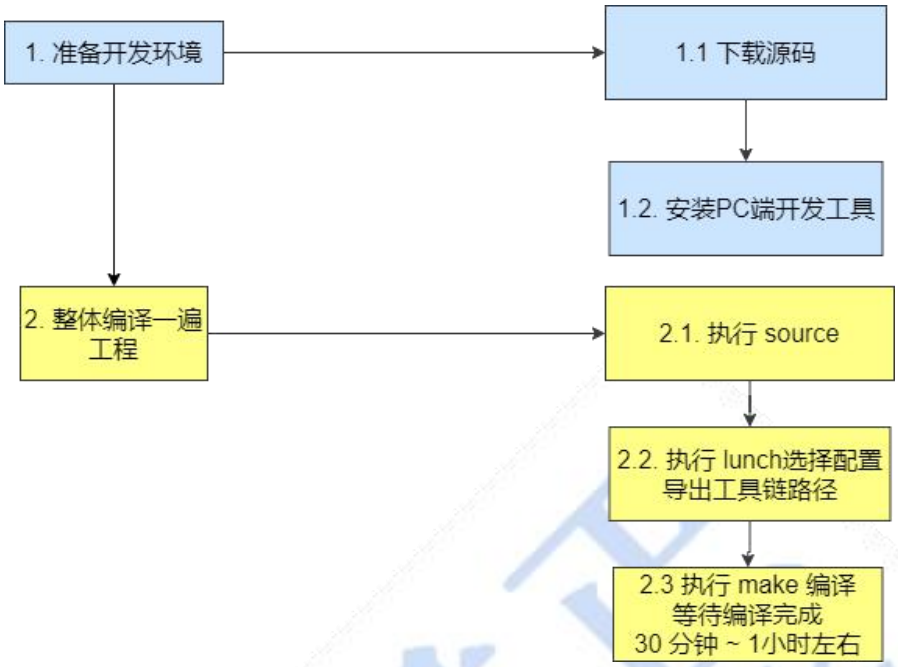
图表 1- 1 内核驱动适配结构图

1.2 内核开发流程

内核的开发可以分为两种类型的开发，一种基于君正提供的 SDK 开发环境进行开发，另外一种类型的是基于源码目录开发流程。

1.2.1 基于君正 SDK 内核开发流程（推荐使用）

通常情况下，会下载完整的 SDK 包，首先按照进行基础开发环境的搭建，整体编译一次（注:这点很重要）。此过程会对内核进行基础的配置。流程图如下:



图表 1- 2 内核开发流程图

在保证以上工作都执行完成之后，在需要重新配置内核，或者修改了内核需要重新编译的情况下，可以在 SDK 顶层执行以下命令进行配置

```
$ make kernel-menuconfig  # 配置内核
$ make kernel             # 编译内核
```

执行该命令后，会生成文件 out/product/productName/image/kernel， 该文件就是最终需要烧录的内核镜像。

基于君正 SDK 的开发流程，源码和编译生成的中间文件是分离开的，中间文件会生成在 out/product/productName/obj/kernel-intermediate 目录下，包括内核的.config 文件。

注意事项：

1. 使用君正 SDK 的开发流程，需要保证源码目录没有被污染，即不能在源码目录下使用传统的 make xx_defconfig 进行内核配置和编译，否则会报错误。此时可以在 kernel 源码目录下执行

```
$ make distclean
$ make mrproper
```

2. 所有使用 make kernel-xxx 的命令，实质执行的是：

```
$ make -C out/product/productName/obj/kernel-intermediate xxx
```

在实际开发过程中，可以尝试将 make kernel-xxx 中 xxx 替换为内核支持的 make 命令

1.2.2 基于内核源码开发流程

基于内核源码的开发流程则是修改和编译的目标文件都生成在 kernel 目录下。该开发流程所有命令的执行都在 kernel 目录下。

```
$ cd kernel
```

```
$ make board_cfg_defconfig
$ make uImage

当需要修改内核重新配置时，使用
$ make menuconfig
$ make uImage

$ ls arch/mips/boot/uImage 即为生成的目标文件
```

执行以上命令后会在内核源码目录下生成 arch/mips/boot/uImage 文件，该文件即为最终烧录的镜像。其中 board_cfg 可以在 arch/mips/configs/目录下找到。

1.3 内核默认配置

所有针对开发板的内核配置默认配置文件都存储在 arch/mips/configs/目录下。

Halley5 所包含的配置文件说明如下:

表格 1 Halley5 内核配置表

配置文件	说明
halley5_v20_linux_msc_defconfig	用于主存储为 SD 卡/eMMC 启动的配置
halley5_v20_linux_sfc_nand_defconfig	用于主存储为 SPI-Nand 的启动配置
halley5_v20_linux_sfc_nor_defconfig	用于主存储为 SPI-Nor 的启动配置
halley5_v20_linux_sfc_nand_recovery_defconfig	用于 OTA 升级的 recovery 配置。
halley5_v20_linux_msc_burn_defconfig	用于 SD 卡烧工具的制作，开发请忽略。
halley5_v20_linux_msc_ltp_defconfig	内部使用，开发请忽略。

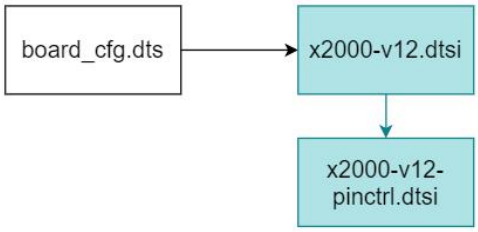
1.4 设备树配置

devicetree 用于描述芯片和开发板板级资源，主要有两种使用方式，一种方式直接和内核编译在一起，另外一种方式单独编译，单独存储，由 uboot 传参告知 dtb 存储的位置。

默认使用直接编译内核的方式，具体区别可以参考下文。

1.4.1 设备树文件介绍

关于设备树的配置文件，主要存放路径为 arch/mips/boot/dts/ingenic 目录，dts 的组成一般是 soc 级定义和 board 级的定义。



图表 1- 3 dts 结构示意图

Halley5 平台的涉及到的 devicetree 文件如下表:

dts 文件	文件说明
halley5_v10.dts	针对 halley5 V1.x 系列的开发板板级配置。自定义开发板配置时，需要修改此文件
halley5_v20.dts	针对 halley5 V2.x 系列的开发板板级配置。自定义开发板配置时，需要修改此文件
halley5_cameras ├── RD_X2000_HALLEY5_CAMERA_1V0.dtsi ├── RD_X2000_HALLEY5_CAMERA_2V0.dtsi ├── RD_X2000_HALLEY5_CAMERA_2V1_cim.dtsi ├── RD_X2000_HALLEY5_CAMERA_2V1.dtsi ├── RD_X2000_HALLEY5_CAMERA_3V2_4LANE.dtsi ├── RD_X2000_HALLEY5_CAMERA_3V2.dtsi ├── RD_X2000_HALLEY5_CAMERA_4V2.dtsi ├── RD_X2000_HALLEY5_CAMERA_4V3.dtsi ├── RD_X2000_HALLEY5_CAMERA_5V0_cim.dtsi ├── RD_X2000_HALLEY5_CAMERA_5V0.dtsi └── RD_X2000_HALLEY5_CAMERA_CIM_BT656.dtsi	此文件夹下的内容针对的是 HALLEY5 支持的 Camera 子板配置，如果需要添加新的 camera 配置，可以参考里面的内容进行添加。如果完全使用官方的配置，不需要修改内容
x2000-v12.dtsi	X2000 SOC 芯片基础配置，客户一般不需要修改。
x2000-v12-pinctrl.dtsi	X2000 GPIO function 配置，客户一般不需要修改。

1.4.2 内核 Builtin 设备树（默认使用方式）

参考平台默认使用的方式。

1.4.3 单独编译设备树

单独编译和使用设备树需要多方面的支持

1. 修改 u-boot，使其支持加载 devicetree。
2. 修改内核配置，使其支持单独编译出 devicetree 二进制文件。
3. 修改烧录工具，使其支持烧录 devicetree 二进制文件。

1.4.3.1 修改 uboot 配置

1. 需要修改配置文件， 如： u-boot/include/configs/halley5.h,

   a) 添加 devicetree 支持

```
/* Device Tree Configuration*/
#define CONFIG_OF_LIBFDT 1
#define IMAGE_ENABLE_OF_LIBFDT 1
#define CONFIG_LMB
```

   b) 修改内核启动参数和启动命令

- Nand 启动:

```
#define CONFIG_BOOTARGS BOOTARGS_COMMON "ip=off init=/linuxrc ubi.mtd=3 root=ubi0:rootfs ubi.mtd=4
rootfstype=ubifs rw
#define CONFIG_BOOTCOMMAND "set uImage 0x80600000; set dtb 0x83000000; sfcnand read 0x100000 0x400000
$(uImage); sfcnand read 0x900000 0x20000 $(dtb); bootm $(uImage) - ${dtb}"
```

- Emmc/SD 卡启动:

```
#define CONFIG_BOOTARGS BOOTARGS_COMMON " rootfstype=ext4 root=/dev/mmcblk0p7 rootdelay=3 rw"
#define CONFIG_BOOTCOMMAND "set dtb 0x83000000; set uImage 0x80f00000; mmc dev 0;mmc read ${uImage}
0x1800 0x2000; mmc read ${dtb} 0x5800 0x100; bootm ${uImage} - ${dtb}"
```

   注意： 需要根据实际分区情况进行修改

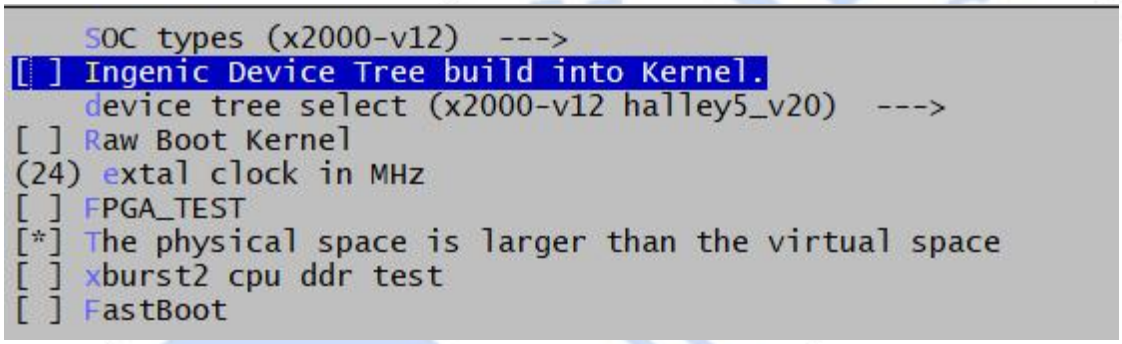
修改内核配置

在 SDK 顶层，执行 make kernel-menuconfig，去掉 INGNEIC_BUILTIN_DTB 配置。

其配置说明如下：

```
There is no help available for this option.
  Symbol: INGENIC_BUILTIN_DTB [=y]
  Type  : boolean
  Prompt: Ingenic Device Tree build into Kernel.
    Location:
      -> Machine selection
        -> SOC Type Selection
    Defined at arch/mips/xburst2/Kconfig:49
    Depends on: MACH_XBURST2 [=y]
    Selects: BUILTIN_DTB [=y]
```

去掉选项之后界面如下：



图表 1- 4 去掉 INGENIC_BUILTIN_DTB

1.4.3.2 编译 dtb 文件

在 SDK 顶层执行

```
$make kernel-dtbs
```

会在out/product/halley5/obj/kernel-intermediate/下生成文件

```
arch/mips/boot/dts/ingenic/halley5_v20.dtb
```

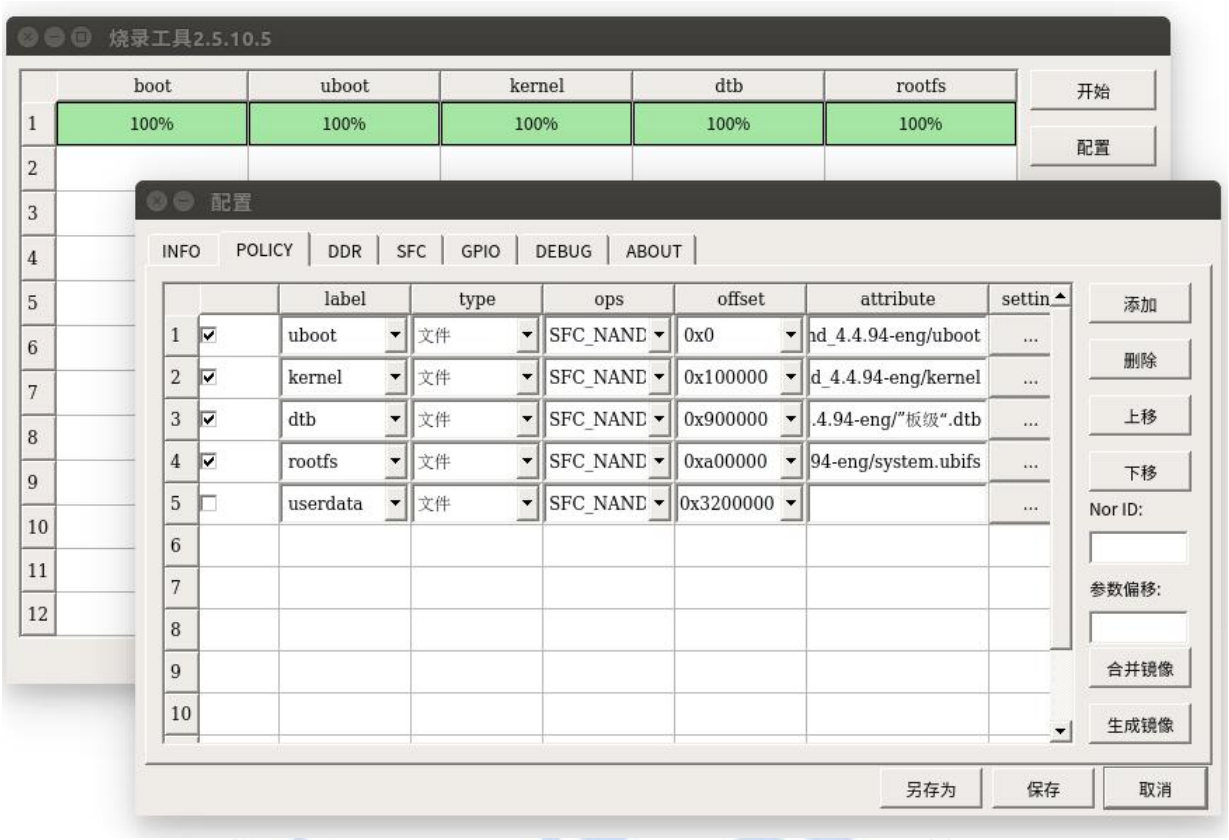
该文件即为最终需要烧写devicetree二进制文件。

1.4.3.3 烧录 dtb 文件

针对 Nand 主存储和 eMMC/SD 主存储不同，烧录工具的配置有区别。

1. **烧录 Nand DTB 文件**

注意：将 dtb文件烧录到设备树预留的分区上， 要和 bootcmd 和 bootargs 配置的保持一致



图表 1- 5 Nand DTB 烧录

2. 烧录 eMMC/SD DTB 文件

a) dtb 存放位置：推荐使用默认地址 0xb00000（11MB），无需做其他修改，直接烧录即可。

b) 如果需要自己指定 dtb 分区，需要做如下修改：

i. 以 halley5 为例，需要参考文件 u-boot/board/ingenic/”板级”/partitiongs.tab

ii. 内核需要修改分区个数，与上面 partitions.tab 配置相同

iii. 修改烧录工具 dtb 位置

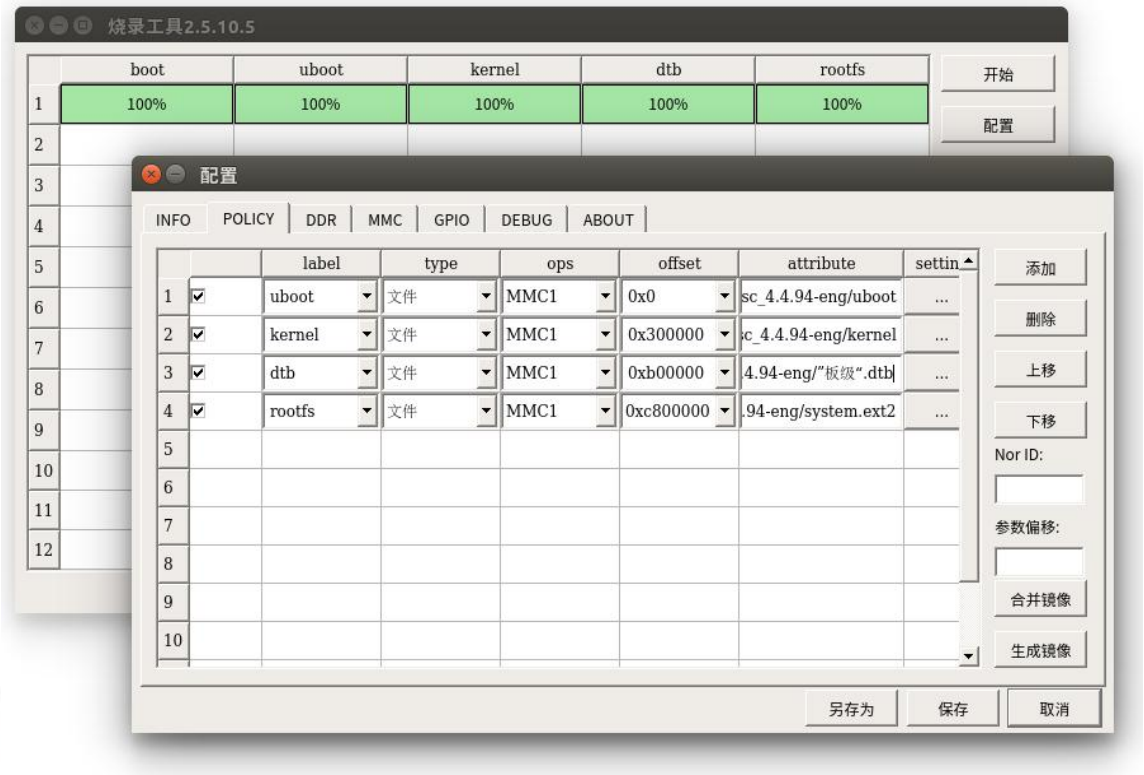
修改 u-boot/board/ingenic/"板级"/partitions.tab

```
property:
    disk_size = 4096m
    gpt_header_lba = 512
    custom_signature = 0

partition:
        #name     =  start,    size, fstype
        xboot     =     0m,     3m,
        boot      =     3m,     8m, EMPTY
        recovery  =    12m,    16m, EMPTY
        pretest   =    28m,    16m, EMPTY
        reserved  =    44m,    52m, EMPTY
```

```
        misc        =    96m,      4m, EMPTY
        cache       =   100m,    100m, LINUX_FS
        system      =   200m,   1800m, LINUX_FS
        data        =  2000m,   2048m, LINUX_FS
```

```
#fstype could be: LINUX_FS, FAT_FS, EMPTY
```

注： dtb 推荐使用地址 0xb00000， 为 MMC 的 11MB 位置



图表 1- 6 eMMC/SD DTB 烧录

1.4.4 通过设备树传递内核参数

内核 bootargs 的传递有多种方式，这里只介绍如何使用 dts 文件传递内核参数。

1.4.4.1修改设备树

定义 chosen 节点，节点中的 bootargs 作为向内核传递的 cmdline。

例： arch/mips/boot/dts/ingenic/halley5_v20.dts

```
/ {
        compatible = "ingenic,halley5", "ingenic,x2000-v12";
        chosen {
                bootargs = "console=ttyS1,115200 mem=128M@0x0ip=off init=/linuxrc ubi.mtd=3 root=ubi0:rootfs
ubi.mtd=4 rootfstype=ubifs rw";
        };
```

```
};
```

1.4.4.2 修改内核配置

内核选择 MIPS_CMDLINE_FROM_DTB 配置后，会解析 dts 文件中的 bootargs 而忽略从 uboot 中传递的 bootargs。

注：启动过程中，uboot 解析 dtb 中的 chosen 节点，如果没有定义，则 uboot 会创建默认的 chosen 节点.

```
Symbol: MIPS_CMDLINE_FROM_DTB [=y]
Type : boolean
Prompt: Dtb kernel arguments if available
  Location:
    -> Kernel type
      -> Kernel command line type (<choice> [=y])
  Defined at arch/mips/Kconfig:2858
  Depends on: <choice> && USE_OF [=y]
```

选中后的图像参考下图

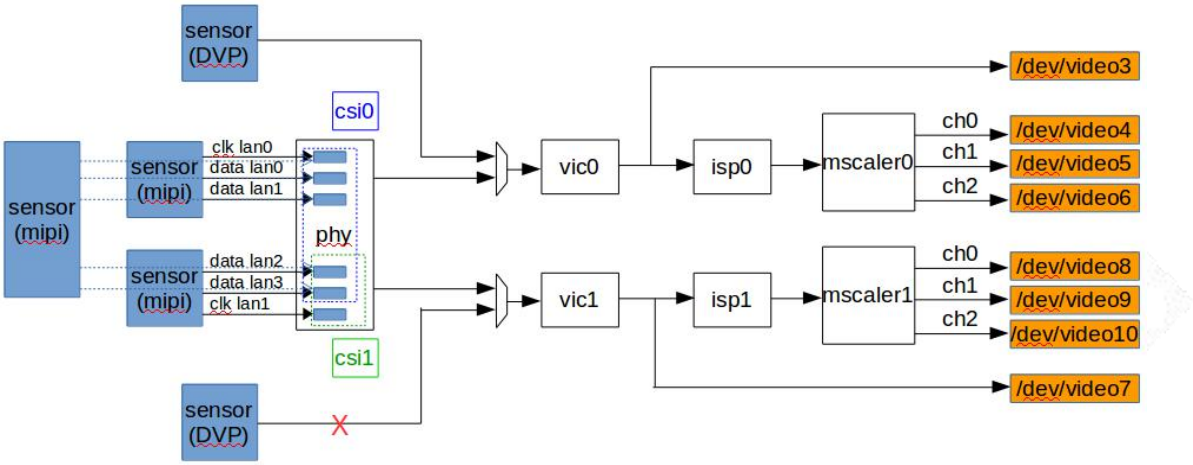
```
[ ] Enable seccomp to safely compute untrusted bytecode
-*- Support for O32 binaries using 64-bit FP
    Kernel appended dtb support (None)  --->
    Kernel command line type (Dtb kernel arguments if available)  --->
```

图表 1- 7

2 ISP 图像处理单元

2.1 模块功能介绍

X2000 芯片包含两个独立的 ISP（ImageSignalProcessing）模块，每个 ISP 模块包含 CSI，VIC，ISP-CORE，MSCALER 四个子模块（以下用编号 0，1 区分），主要功能是对前端传感器输出的图像信号进行处理。硬件拓扑如下图所示。



图表 2- 1 ISP 模块硬件拓扑图

支持 DVP，MIPI，BT656 输入，VIC 支持 RAW，YUV422 格式输出，MSCALER 支持 NV12，NV21 格式缩放输出。ISP-CORE 能对图像信号进行黑电平矫正，坏点矫正，镜头阴影矫正,自动白平衡，自动曝光，自动增益,GAMMA 曲线矫正，CCM 矫正，2D 降噪等处理。

2.2 驱动源码位置

驱动源码所在位置：

```
drivers/media/platform/ingenic-isp
├──csi.c
├──csi-regs.h
├──isp.c
├──isp-core
| ├──inc
| ├──system_sensor_drv.h
| ├──tiziano_core.h
| ├──tiziano_core_tuning.h
| ├──tiziano_isp.h
| └──tiziano_sys.h
| ├──isp-core.a
| ├──Makefile
├──isp-drv.c
├──isp-drv.h
```

```
├──isp-regs.h
├──isp-video.c
├──isp-video-mplane.c
├──Makefile
├──mscaler.c
├──mscaler-regs.h
├──sensor.c
├──vic.c
└──vic-regs.h
```

2.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

ISP 控制器描述：

```
ispcam0:
        isp-camera@0 {
            compatible = "ingenic,x2000-isp-camera";
             ......
        };

        csi0: csi@0x10074000 {
            compatible = "ingenic,x2000-csi";
            ......
        };

        vic0: vic@0x13710000 {
            compatible = "ingenic,x2000-vic";
           ......
        };
        isp0: isp@0x13700000 {
            compatible = "ingenic,x2000-isp";
            ......
        };
        mscaler0: mscaler@0x13702300 {
            compatible = "ingenic,x2000-mscaler";
           ......;
        };
    };

ispcam1: isp-camera@1 {
        ......
```

```
};
```

2.3.1 设备树默认配置

设备树默认配置产生 ISP 设备，支持外接双摄摄像头子板 RD_X2000_HALLEY5_CAMERA_V4.2。

2.3.1.1 isp 控制器配置

在板级设备树 halley5_v20.dts 中，对 isp-dp 进行如下默认配置：

```
&isp0_ep {
        remote-endpoint = <&ov2735_ep0>;
        bus-width = <10>;       /* Used data lines */
        data-shift = <0>;       /* Lines 9:0 are used */

        /* If hsync-active/vsync-active are missing,
           embedded BT.656 sync is used */
        hsync-active = <1>;     /* Active high */
        vsync-active = <1>;     /* Active high */
        data-active = <1>;      /* Active high */
        pclk-sample = <1>;      /* Rising */
};

&isp1_ep {
        remote-endpoint = <&ov2735_ep1>;
        data-lanes = <3 4>;
        clk-lanes = <5>;
};
```

2.3.1.2 camera sensor 配置

在 halley5_cameras/RD_X2000_HALLEY5_CAMERA_4V2.dtsi 中，对 camera sensor 进行如下默认配置：

```
&i2c3 {
        status = "okay";
        clock-frequency = <100000>;
        timeout = <1000>;
        pinctrl-names = "default";
        pinctrl-0 = <&i2c3_pa>;

        /*RD_X2000_HALLEY5_CAMERA_V4.2 DVP interface*/
        ov2735_0:ov2735@0x3d {
                status = "ok";
                compatible = "ovti,ov2735b";
                reg = <0x3d>;
                pinctrl-names = "default", "default";
                pinctrl-0 = <&vic_pa_low_10bit>;
```

```
                pinctrl-1 = <&cim_vic_mclk_pe>;

                ingenic,rst-gpio = <&gpa 10 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,ircutp-gpio = <&gpb 3 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
                ingenic,ircutn-gpio = <&gpb 0 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;

                port {
                        ov2735_ep0:endpoint {
                                remote-endpoint = <&isp0_ep>;
                                bus-width = <10>;       /* Used data lines */
                                data-shift = <0>;       /* Lines 9:0 are used */

                                /* If hsync-active/vsync-active are missing,
                                   embedded BT.656 sync is used */
                                hsync-active = <1>;     /* Active high */
                                vsync-active = <1>;     /* Active high */
                                data-active = <1>;      /* Active high */
                                pclk-sample = <1>;      /* Rising */
                        };
                };
        };


        /*RD_X2000_HALLEY5_CAMERA_V4.2 MIPI interface*/
        ov2735_1:ov2735@0x3c {
                status = "ok";
                compatible = "ovti,ov2735a";
                reg = <0x3c>;
                pinctrl-names = "default";
                pinctrl-0 = <&cim_vic_mclk_pe>;

                ingenic,rst-gpio = <&gpa 11 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,ircutp-gpio = <&gpb 7 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
                ingenic,ircutn-gpio = <&gpb 1 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;

                port {
                        ov2735_ep1:endpoint {
                                remote-endpoint = <&isp1_ep>;
                        };
                };
        };
};
```

2.3.2 设备树自定义配置

1.设备树可选配 halley5 适配的其他 camera 小板，选配 halley5 适配的其他 camera 小板时，在板级.dts 中 includle arch/mips/boot/dts/ingenic/halley5_cameras 目录中相应的 dtsi 即可，该配置可通过 menuconfig 完成。

2.设备树可选配用户自定义的 camera sensor，选配用户自定义的 camera sensor 时，仿照默认配置，根据接口类型进行 camera sensor 及 isp end point 的配置。

2.4 内核编译配置

内核配置 VIDEO_INGENIC_ISP，配置说明如下：

```
Symbol: VIDEO_INGENIC_ISP [=y]
Type  : tristate
Prompt: V4L2 Driver for ingenic isp
        Location:
            -> Device Drivers
                -> Multimedia support (MEDIA_SUPPORT [=y])
                     -> V4L platform devices (V4L_PLATFORM_DRIVERS [=y])Kconfig:117
        Depends on: MEDIA_SUPPORT [=y] && V4L_PLATFORM_DRIVERS [=y] && VIDEO_DEV [=y] && VIDEO_V4L2 [=y]
        Selects: VIDEOBUF2_DMA_CONTIG [=y]


Symbol: VIDEO_V4L2_SUBDEV_API [=y]
Type  : boolean
Prompt: V4L2 sub-device userspace API
        Location:
            -> Device Drivers
                -> Multimedia support (MEDIA_SUPPORT [=y])
        Defined at drivers/media/Kconfig:117
        Depends on: MEDIA_SUPPORT [=y] && VIDEO_DEV [=y] && MEDIA_CONTROLLER [=y]
```

2.4.1 内核默认编译配置

内核默认配置 ISP 驱动，并支持 RD_X2000_HALLEY5_CAMERA_V4.2，配置界面如下：

```
--- V4L platform devices
<*>   V4L2 Driver for ingenic isp
[*]     vic dma out route enable
          vic dma out route select (vic dma bebug sys node)  --->
[*]     halley5_camera_board  --->
[ ]     gewu_camera_board  ----
< >   SoC camera support
< >   Xilinx Video IP (EXPERIMENTAL)
```

```
--- halley5_camera_board
[ ]   halley5 camera driver for RD_X2000_HALLEY5_CAMERA_1V0
[ ]   halley5 camera driver for RD_X2000_HALLEY5_CAMERA_2V1
[*]   halley5 camera driver for RD_X2000_HALLEY5_CAMERA_4V2
[ ]   halley5 camera driver for RD_X2000_HALLEY5_CAMERA_4V3
[ ]   halley5 camera driver for RD_X2000_HALLEY5_CAMERA_3V2
[ ]   halley5 camera driver for RD_X2000_HALLEY5_CAMERA_5V0
```

2.4.2 内核自定义编译配置

2.4.2.1 Camera 子板编译配置

选配 halley5 适配的其他 camera 小板，修改默认配置中的 halley5_camera_board 选项。

2.4.2.2 配置 debug 调试接口

用于打开 VIC_DEBUG 功能。

```
Symbol : VIC_DMA_DEBUG[=y]
Type : Boolean
Prompt : vic dma bebug sys node
  Location :
    ->Device Drivers
      ->Multimedia support (MEDIA_SUPPORT[=y])
        ->V4L platform devices (V4L_PLATFORM_DRIVERS[=y])
          ->V4L2 Driver for ingenic isp (VIDEO_INGENIC_ISP[=y])
            ->vic dma out route enable (VIC_DMA_ROUTE[=y])
              ->vic dma out route select (<choice>[=y])
  Defined at drivers/media/platform/ingenic-isp/Kconfig:16
  Dependson : <choice> && VIC_DMA_ROUTE [=y]
```

2.4.2.3 配置自定义 camera sensor

当 ISP 连接用户自定义的 camera sensor 时，可仿照已有 sensor 型号配置 sensor drivers，代码所在位置：

```
drives/meida/i2c/ingenic-isp/
```

配置界面如下：

```
   *** ingenic-isp camera sensor drivers ***
< > ov4689 camera support
< > sc2232h camera support
-*- ov2735 camera DVP interface support
-*- ov2735 camera MIPI interface support
< > ar0144 camera support
< > ar0234 camera support
< > ov7251 camera MIPI interface support
< > sc031gs camera MIPI interface support
< > ov6710 camera MIPI interface support
```

2.5 设备节点生成

驱动加载成功后生成以下节点：

2.5.1 Debug 节点

```
/sys/devices/platform/ahb0/ahb0:isp-camera@0/10074000.csi/debug/dump_csi
```

csi0 debug 节点，用于打印 csi0 相关寄存器状态

```
/sys/devices/platform/ahb0/ahb0:isp-camera@0/13710000.vic/debug/dump_vic
```

vic0 debug 点，用于打印 vic0 相关寄存器状态

```
/sys/devices/platform/ahb0/ahb0:isp-camera@0/13710000.vic/debug/vic_dma_debug
```

Vic0 debug 节点，配置 VIC_DMA_DEBUG 时生成，用于在通过 mscaler0 video 节点收图时，获取一帧未经 isp 处理的 raw 图。

/sys/devices/platform/ahb0/ahb0:isp-camera@0/13700000.isp/debug/dump_isp

isp0 debug 节点，用于打印 isp-core0 相关寄存器状态

/sys/devices/platform/ahb0/ahb0:isp-camera@0/13702300.mscaler/debug/dump_mscaler

mscaler0 debug 节点，用于打印 mscaler0 相关寄存器状态

/sys/devices/platform/ahb0/ahb0:isp-camera@1/10073000.csi/debug/dump_csi

csi1 debug 节点，用于打印 csi1 相关寄存器状态

/sys/devices/platform/ahb0/ahb0:isp-camera@1/13810000.vic/debug/dump_vic

vic1 debug 节点，用于打印 vic1 相关寄存器状态

/sys/devices/platform/ahb0/ahb0:isp-camera@1/13810000.vic/debug/vic_dma_debug

vic1 debug 节点，配置 VIC_DMA_DEBUG 时生成，用于在通过 mscaler1 video 节点收图时，获取一帧未经 isp 处理的 raw 图。

/sys/devices/platform/ahb0/ahb0:isp-camera@1/13800000.isp/debug/dump_isp

isp1 debug 节点，用于打印 isp-core1 相关寄存器状态

/sys/devices/platform/ahb0/ahb0:isp-camera@1/13802300.mscaler/debug/dump_mscaler

mscaler1 debug 节点，用于打印 mscaler1 相关寄存器状态

2.5.2 Video 节点

/dev/video3

vic0 设备节点，vic0 输出节点的实例化，用于输出未经 isp 处理的 raw 数据或 yuv422 数据

/dev/video4

mscaler0-ch0 设备节点，mscaler0-ch0 输出节点的实例化，用于输出经 isp 处理并经 mscaler channel0 缩放的 NV12 或 NV21 数据

/dev/video5

mscaler0-ch1 设备节点（默认不生成，修改驱动中 MSCALER_MAX_CH = 2 or 3 时生成），mscaler0-ch1 输出节点，用于输出经 isp 处理并经 mscaler channel1 缩放的 NV12 或 NV21 数据

/dev/video6

mscaler0-ch2 设备节点 （默认不生成，修改驱动中 MSCALER_MAX_CH = 3 时生成），mscaler0-ch2 输出节点，用于输出经 isp 处理并经 mscaler channel2 缩放的 NV12 或 NV21 数据

/dev/video7

vic1 设备节点，vic1 的输出节点，用于输出未经 isp 处理的 raw 数据或 yuv422 数据

/dev/video8

mscaler1-ch0 设备节点，mscaler1-ch0 的输出节点，用于输出经 isp 处理并经 mscaler channel0 缩放的 NV12 或 NV21 数据

/dev/video9

mscaler1-ch1 设备节点（默认不生成，修改驱动中 MSCALER_MAX_CH = 2 or 3 时生成），mscaler1-ch1 的输出节点，用于输出经 isp 处理并经 mscaler channel1 缩放的 NV12 或 NV21 数据

/dev/video10

mscaler1-ch2 设备节点 （默认不生成，修改驱动中 MSCALER_MAX_CH = 3 时生成），mscaler1-ch2

的输出节点，用于输出经 isp 处理并经 mscaler channel2 缩放的 NV12 或 NV21 数据

2.6 应用程序使用说明

2.6.1 v4l2-ctl

v4l2-ctl 可测试 isp 通路（vic 节点输出或 mscaler 节点输出）并保存图像数据到文件。

2.6.1.1 源码位置

```
buildroot/dl/libv4l/v4l-utils-1.18.0.tar.bz2
```

2.6.1.2 命令行及参数示意

```
v4l2-ctl -v width=640, height=480, pixelformat="NV12" --stream-mmap=3 --stream-to="test-ch0.yuv" -d /dev/video3
```

- width 指定输出图像宽度；
- heigh 指定输出图像高度；
- pixelformat 指定输出图像格式；
- --stream-mmap 指定轮转 buffer 数量；
- --stream-to 指定储存图像文件名；
- -d 指定图像输出的 video 节点

注意，当使用 vic 的 video 节点输出图像时（即 video3，video7），width，height 须指定为图像的原始宽高,pixelformat 需根据 sensor 输出格式指定为执行 v4l2-ctl –list-format –d /dev/video3 命令后输出的格式；

当使用 mscaler 的 video 节点输出图像时（即 video4，video5，video6，video8，video9，video10），width，height 可指定图像缩放的宽高，pixelformat 支持 NV12，NV21。

2.6.2 ffmpeg

ffmpeg 可测试 isp 通路（mscaler 节点输出）并在 LCD 屏上预览。

2.6.2.1 源码位置

```
buildroot/dl/ffmpeg/ffmpeg-4.2.2.tar.xz
```

2.6.2.2 命令行及参数示意

```
echo 6 > /sys/devices/platform/ahb0/13050000.dpu/layer0/src_fmt
echo 1 > /sys/devices/platform/ahb0/13050000.dpu/comp_update
```

设置 layer0 层的显示格式为 NV12

```
ffmpeg -pix_fmt nv12 -s 640*480 -i /dev/video4 -f fbdev /dev/fb0
```

- -pix_fmt 指定输出格式，支持 nv12 或 nv21；
- -s 指定输出图像大小，可缩放
- -i 指定输出节点，支持 mscaler video 节点（即 video4，video5，video6，video8，video9，video10）
- -f 指定输出设备节点

2.6.3 cimutils

测试 isp 通路（mscaler 节点输出）并对图像进行硬件编码

2.6.3.1 源码位置

```
packages/example/App/cimutils
```

2.6.3.2 命令行及参数示意

```
cimutils -v isp - E helix -C -f 50.jpg -t nv12 -x 320 -y 240
```

或

```
cimutils -I 1 -I 4 -C -f 40.jpg -t nv12 -x 320 -y 240
```

- -x 指定图像宽度；
- -y 指定图像高度；
- -t 指定图像格式；
- -f 指定文件名；
- -v 指定 isp 控制器
- -E 指定 helix 硬件编码
- -I 指定 helix 对应的 video 节点号(video1)和 mscaler 对应的 video 节点号(video4,video5，video6，video8，video9，video10）

2.6.4 v4l2-isp-tuning

isp-core 提供亮度，对比度，饱和度，锐度的调节接口，可通过 v4l2-isp-tuning 操作 mscaler 的某一节点对以上参数进行调节，并通过 mscaler 的另一节点观察调节效果。或参考源码中 API 的使用方法合并到用户代码中。

2.6.4.1 源码位置

```
packages/example/App/v4l2-isp-tuning
```

2.6.4.2 命令行及参数示意

```
v4l2-isp-tuning -w 640 -h 480 -i 4 -s 128 -c 128 -S 128 -b 128
```

- -w 指定图像宽度；
- -h 指定图像高度；
- -i 指定用于调节效果参数的 mscaler 节点号（video4，video5，video6，video8，video9，video10）；
- -s 指定锐度，最小值 0，最大值 255，默认值 128；
- -c 指定对比度，最小值 0，最大值 255，默认值 128；
- -S 指定饱和度，最小值 0，最大值 255，默认值 128；
- -b 指定亮度，最小值 0，最大值 255，默认值 128；

3 CIM 摄像头接口模块

3.1 模块功能介绍

介绍模块的基本功能，驱动实现的功能，驱动实现的架构和框架图。默认配置的功能，需要自己配置的功能等内容。

CIM（camera interface module）模块实现接受前端 camera sensor 发送的图像信号，支持 8bit DVP，MIPI，BT656 数据输入，支持 YUV422，RGB888，RGB565，MONO(8)等格式，支持 snapshot 功能。

3.2 驱动源码位置

驱动源码位于：

```
drivers/media/platform/soc_camera/ingenic/cim-v2
├──ingenic_camera.c
├──ingenic_camera.h
├──Kconfig
├──Makefile
├──mipi_csi.c
└──mipi_csi.h
```

3.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dts
```

CIM 控制器描述：

```
cim: cim@0x13060000 {
        compatible = "ingenic,x2000-cim";
        reg = <0x13060000 0x10000>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_CIM>;
        clocks = <&clock CLK_DIV_CIM>, <&clock CLK_GATE_CIM>, <&clock CLK_GATE_MIPI_CSI>;
        clock-names = "div_cim", "gate_cim", "gate_mipi";
        status = "disable";
};
```

3.3.1 设备树默认配置

设备树默认编译不会产生 CIM 设备。

3.3.2 设备树自定义配置

以 CIM 连接 AR0144 sensor 为例进行自定义配置。

3.3.2.1 CIM 控制器配置

在 halley5_cameras/RD_X2000_HALLEY5_CAMERA_5V0_cim.dtsi 中，对 cim 控制器进行如下自定义

配置：

```
&cim {
        status = "okay";
        port {
                cim_0: endpoint@0 {
                        remote-endpoint = <&AR0144_0>;
                        bus-width = <8>;
                };
        };
};
```

3.3.2.2 camera sensor 配置

在 halley5_cameras/RD_X2000_HALLEY5_CAMERA_5V0_cim.dtsi 中，对 camera sensor 进行如下自定义配置：

```
&i2c3 {
        status = "okay";
        clock-frequency = <100000>;
        timeout = <1000>;
        pinctrl-names = "default";
        pinctrl-0 = <&i2c3_pa>;

        /*RD_X2000_HALLEY5_CAMERA_V3.2*/
        AR0144:AR0144@0x18 {
                status = "ok";
                compatible = "onsemi,ar0144";
                reg = <0x18>;
                pinctrl-names = "default","cim";
                pinctrl-0 = <&cim_vic_mclk_pe>, <&cim_pa>;

                resetb-gpios = <&gpa 11 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                pwdn-gpios = <&gpa 10 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;

                port {
                        AR0144_0:endpoint {
                                remote-endpoint = <&cim_0>;
                        };
                };
        };
};
```

3.4 内核编译配置

内核配置 VIDEO_INGENIC_CIM_V2,配置说明如下：

```
Symbol: VIDEO_INGENIC_CIM_V2 [=n]
```

```
Type  : tristate
Prompt: Ingenic Soc Camera Driver for X2000 && M300
  Location:
    -> Device Drivers
      -> Multimedia support (MEDIA_SUPPORT [=y])
        -> V4L platform devices (V4L_PLATFORM_DRIVERS [=y])
          -> Ingenic Camera Sensor Interface driver (VIDEO_INGENIC [=n])
  Defined at drivers/media/platform/soc_camera/ingenic/cim-v2/Kconfig:1
  Depends on: MEDIA_SUPPORT [=y] && V4L_PLATFORM_DRIVERS [=y] && VIDEO_INGENIC [=n]
```

3.4.1 内核默认编译配置

内核默认未编译 CIM 驱动。

3.4.2 内核自定义编译配置

3.4.2.1 CIM 控制器配置

打开 CIM 控制器驱动，配置界面如下：

```
--- V4L platform devices
< >   V4L2 Driver for ingenic isp
<*>   SoC camera support
< >   platform camera support (NEW)
<*>   Ingenic Camera Sensor Interface driver
< >     Ingenic Soc camera Driver for X1000 (NEW)
<*>     Ingenic Soc Camera Driver for X2000 && M300
[ ]       Sensor support snapshot function (NEW)
< >   Xilinx Video IP (EXPERIMENTAL)
```

使用 snapshot 功能时打开 Sensor support snapshot function，并配置脉冲宽度和 delay 时间。

3.4.2.2 camera sensor 配置

选择相应 sensor 驱动，以 AR0144 为例，配置界面如下：

```
    *** soc_camera sensor drivers ***
< > ov5640 camera support (NEW)
< > gc2155 camera support (NEW)
< > imx074 support (NEW)
< > mt9m001 support (NEW)
< > mt9m111, mt9m112 and mt9m131 support (NEW)
< > mt9t031 support (NEW)
< > mt9t112 support (NEW)
< > mt9v022 and mt9v024 support (NEW)
< > ov2640 camera support (NEW)
< > ov5642 camera support (NEW)
< > ov5645 camera support (NEW)
< > ov9281 camera support (NEW)
< > ov6650 sensor support (NEW)
< > ov772x camera support (NEW)
< > ov7725 camera support (NEW)
< > ov9640 camera support (NEW)
< > ov9740 camera support (NEW)
< > rj54n1cb0c support (NEW)
< > tw9910 support (NEW)
<*> ar0144 support
< > ov2735b support (NEW)
```

使用 snapshot 功能时，在 sensor 驱动中将 sensor 配置为 sanpshot 模式。

3.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/video11
```

cim 的输出节点。

3.6 应用程序使用说明

3.6.1 cimutils

3.6.1.1 源码位置

```
packages/example/App/cimutils
```

3.6.1.2 命令行及参数示意

```
cimutils -C -v cim -x 320 -y 240 -t grey -f 1.raw -l 0
```

- -x 指定图像宽度；
- -y 指定图像高度；
- -v 指定 cim 控制器；
- -t 指定图像格式；
- -f 指定图像文件名。

3.6.2 v4l2-ctl

3.6.2.1 源码位置

```
buildroot/dl/libv4l/v4l-utils-1.18.0.tar.bz2
```

3.6.2.2 命令行及参数示意

```
v4l2-ctl  -v  width=640,  height=480,  pixelformat="GREY"  --stream-mmap=3  --stream-to="cim-test.yuv"  -d
/dev/video11
```

- width 指定输出图像宽度；
- heigh 指定输出图像高度；
- pixelformat 指定输出图像格式；
- --stream-mmap 指定轮转 buffer 数量；
- --stream-to 指定储存图像文件名；
- -d 指定图像输出的 video 节点

4 VPU Felix 视频解码处理单元

4.1 模块功能介绍

felix 是 h264 解码,包括:流解析器、运动补偿、反量化、IDCT 和 De-blockengines 的功能。

4.2 驱动源码

驱动源码所在位置：

```
drivers/media/platform/ingenic-vcodec
├── felix
|   ├── felix_drv.c
|   ├── felix_drv.h
|   ├── felix_ops.c
|   ├── felix_ops.h
|   ├── libh264
```

4.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dts
```

felix 控制器描述：

```
felix: felix@0x13300000 {
            compatible = "ingenic,x2000-felix";
            reg = <0x13300000 0x100000>;
            interrupt-parent = <&core_intc>;
            interrupts = <IRQ_FELIX>;
            status = "disabled";
 };
```

4.3.1 设备树默认配置

设备树默认编译会产生 felix 设备：

```
&felix {
      status = "okay";
};
```

4.3.2 设备树自定义配置

用户可根据实际需求关闭 helix 设备，将该节点设置为 disabled：

```
&felix {
      status = "disaled";
};
```

4.4 内核编译配置

内核配置 VIDEO_INGENIC_VCODEC，配置说明如下：

```
Symbol: VIDEO_INGENIC_VCODEC [=y]
Type  : tristate
Prompt: V4L2 driver for ingenic Video Codec
  Location:
    -> Device Drivers
      -> Multimedia support (MEDIA_SUPPORT [=y])
        -> Memory-to-memory multimedia devices (V4L_MEM2MEM_DRIVERS [=y])
  Defined at drivers/media/platform/Kconfig:188
  Depends on: MEDIA_SUPPORT [=y] && V4L_MEM2MEM_DRIVERS [=y] && VIDEO_DEV [=y] && VIDEO_V4L2 [=y]
  Selects: V4L2_MEM2MEM_DEV [=y] && VIDEOBUF2_DMA_CONTIG_INGENIC [=y] && VIDEOBUF2_DMA_CONTIG [=y]
```

4.4.1 内核默认编译配置

内核默认打开 Felix 驱动，配置界面如下：

```
--- Memory-to-memory multimedia devices
< >   Deinterlace support
<*>   Ingenic rotate driver
<*>   V4L2 driver for ingenic Video Codec
< >   SuperH VEU mem2mem video processing driver
```

4.4.2 内核自定义编译配置

用户可根据实际需求去掉该驱动的配置。

4.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/video2
```

felix 的设备节点

4.6 应用程序使用说明

4.6.1 v4l2-h264dec

h264 解码测试

4.6.1.1 源码位置

```
packages/example/App/v4l2-h264dec
```

4.6.1.2 命令行及参数示意

```
v4l2_h264dec -t nv12/nv21/tile420 -v /dev/video2 -f video.mp4.dump.h264 -w 1280 -h 720
```

- -v 指定 felix 的 video 节点；
- -f 指定要解码的码流；
- -w 指定图片的宽度；
- -h 指定图片的高度；

- -s 指定解码后的数据存为图片；
- -p 指定解码后的数据在屏幕预览。

5 VPU Helix 视频编码处理单元

5.1 模块功能介绍

helix 是 h264 编码、JPEG 压缩和解压缩。

5.2 驱动源码位置

驱动源码所在位置：

```
drivers/media/platform/ingenic-vcodec
├── helix
|   ├── api
|   ├── default_sliceinfo.c
|   ├── h264e.c
|   ├── h264e.h
|   ├── h264enc
|   ├── helix_buf.h
|   ├── helix_drv.c
|   ├── helix_drv.h
|   ├── helix_ops.c
|   ├── helix_ops.h
|   ├── jpgd.c
|   ├── jpgd.h
|   ├── jpge
|   ├── jpge.c
|   ├── jpge.h
|   ├── Makefile
|   └── README
```

5.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dts
```

helix 控制器描述：

```
helix: helix@0x13200000 {
            compatible = "ingenic,x2000-helix";
            reg = <0x13200000 0x100000>;
            interrupt-parent = <&core_intc>;
            interrupts = <IRQ_HELIX>;
            status = "disabled";
 };
```

5.3.1 设备树默认配置

设备树默认编译会产生 helix 设备：

```
&helix {
      status = "okay";
};
```

5.3.2 设备树自定义配置

用户可根据实际需求关闭 helix 设备，将该节点设置为 disabled：

```
&helix {
      status = "disaled";
};
```

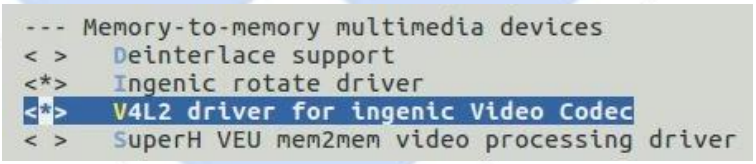
5.4 内核编译配置

内核配置 VIDEO_INGENIC_VCODEC，配置说明如下：

```
Symbol: VIDEO_INGENIC_VCODEC [=y]
Type  : tristate
Prompt: V4L2 driver for ingenic Video Codec
  Location:
    -> Device Drivers
      -> Multimedia support (MEDIA_SUPPORT [=y])
        -> Memory-to-memory multimedia devices (V4L_MEM2MEM_DRIVERS [=y])
  Defined at drivers/media/platform/Kconfig:188
  Depends on: MEDIA_SUPPORT [=y] && V4L_MEM2MEM_DRIVERS [=y] && VIDEO_DEV [=y] && VIDEO_V4L2 [=y]
  Selects: V4L2_MEM2MEM_DEV [=y] && VIDEOBUF2_DMA_CONTIG_INGENIC [=y] && VIDEOBUF2_DMA_CONTIG [=y]
```

5.4.1 内核默认编译配置

内核默认打开 Helix 驱动，配置界面如下：



5.4.2 内核自定义编译配置

用户可根据实际需求去掉该驱动的配置。

5.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/video1
```

helix 的设备节点

5.6 应用程序使用说明

5.6.1 v4l2_h264enc

测试 h264 编码

5.6.1.1 源码位置

```
packages/example/App/v4l2-h264enc
```

5.6.1.2 命令行及参数示意

```
v4l2_h264enc -t nv12 -v /dev/video1 -f video-1280x720_nv12.yuv -w 1280 -h 720
```

- -v 指定 helix 的 video 节点；
- -f 指定要编码的文件；
- -w 指定图片的宽度；
- -h 指定图片的高度。

执行完成后生成 output.h264。

5.6.2 v4l2_jpegdec

测试 jpeg 解码

5.6.2.1 源码路径

```
packages/example/App/v4l2-jpegdec
```

5.6.2.2 命令行及参数示意

```
v4l2_jpegdec -v /dev/video1 -f test.jpg
```

- -v 指定 helix 的 video 节点；
- -f 指定要解码的文件；

执行完成后生成 output.raw。

5.6.3 v4l2_jpegenc

测试 jpeg 编码

5.6.3.1 源码路径

```
packages/example/App/v4l2-jpegenc
```

5.6.3.2 命令行及参数示意

```
v4l2_jpegenc -v /dev/video1 -f video-1280x720_nv12.yuv
```

- -v 指定 helix 的 video 节点；
- -f 指定要编码的文件；

执行完成后生成 output.jpg。

6 Display Controller 显示处理单元

6.1 模块功能介绍

- 图层特性：

显示处理单元支持 4 层 DMA 通道；
输入格式支持 RGB88，ARGB8888，RGB565，RGB555，ARGB1555，NV12/NV21；
支持 2 级 TLB；
支持图像裁剪；

- 组合特性：

支持 4 层透明混合处理；
支持 2 层缩放；
支持写回 DMA；

- 显示特性：

支持 TFT（MIPI-DPI），SLCD（MIPI-DBI type A，B and C），MIPI-DSI；

6.2 驱动源码位置

驱动源码所在位置：

```
drivers/video/fbdev/ingenic/fb_stage
├── displays
|   ├── Kconfig
|   ├── Makefile
|   └── panel-ma0060.c
├── dpu_reg.h
├── ingenicfb.c
├── ingenicfb.h
├── dpu_ctrl.c
├── dpu_ctrl.h
├── jz_mipi_dsi
|     ├── jz_mipi_dsih_hal.c
|     ├── jz_mipi_dsi.c
|     ├── jz_mipi_dsi_lowlwvwl.c
```

6.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/halley5_v20.dts
```

DPU 控制器描述：

```
dpu: dpu@0x13050000 {
        compatible = "ingenic,x2000-dpu";
```

```
        reg = <0x13050000 0x10000>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_LCD>;
        status = "disabled";
};
```

6.3.1 设备树默认配置

设备树默认编译会产生 DPU 控制器设备。

6.3.1.1 DPU 控制器配置

在 halley5_v20.dts 中配置如下：

```
&dpu {
     status="okay";
     port {
          dpu_out_ep : endpoint {
               remote-endpoint=<&panel_ma0060_ep>;
          };
     };
};
```

6.3.1.2 显示屏配置

```
display-dbi {
        compatible = "simple-bus";
        #interrupt-cells = <1>;
        #address-cells = <1>;
        #size-cells = <1>;
        ranges = <>;
        panel_ma0060 {
                compatible = "ingenic,ma0060";
                status = "okay";
                pinctrl-names = "default";
                pinctrl-0 = <&smart_lcd_pb_te>;
                ingenic,vdd-en-gpio = <&gpc 3 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
                ingenic,rst-gpio = <&gpc 4 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,oled-gpio = <&gpc 5 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,lcd-pwm-gpio = <&gpc 1 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,swire-gpio = <&gpc 7 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                port {
                        panel_ma0060_ep: endpoint {
                        remote-endpoint = <&dpu_out_ep>;
                        };
                };
        };
```

```
};
```

6.3.2 设备树自定义配置

用户可根据实际需求关闭 dpu 设备,将以上节点配置为 disabled,或匹配其他型号显示屏。x2000 dpu 支持的屏幕种类：

- smart lcd: dbi 硬件接口，液晶屏有自己的 ram，防裂屏需要有 te 功能
- tft lcd: dpi 硬件接口，没有 ram，需要按照一定的帧率刷新屏幕
- mipi smart lcd: dsi 硬件接口，液晶屏有自己的 ram
- mipi tft lcd: dsi 硬件接口，没有 ram，需要按照一定的帧率刷新屏幕

6.3.2.1 屏幕配置

```
struct lcd_panel {
        const char *name;
        unsigned int num_modes;  /*显示模式支持的数量，固定值 1*/
        struct fb_videomode *modes; /*显示模式*/
        struct jzdsi_data *dsi_pdata;  /*如果屏幕是 mipi dsi 接口需要实现*/

        enum ingenic_lcd_type lcd_type; /*屏幕种类，如果是 mipi tft lcd 需要配置 LCD_TYPE_TFT，其它的按功
能配置*/
        unsigned int bpp;     /*不需要填充*/
        unsigned int width;  /*屏幕实际物理宽度，单位 mm*/
        unsigned int height;  /*屏幕实际物理高度，单位 mm*/

        struct smart_config *smart_config; /*如果屏幕是 smart lcd 需要实现*/
        struct tft_config *tft_config; /*如果屏幕是 tft lcd 需要实现*/

        unsigned dither_enable:1; /*打开 dither 功能*/
        struct {
                unsigned dither_red;
                unsigned dither_green;
                unsigned dither_blue;
        } dither;

        struct lcd_panel_ops *ops; /*不需要实现*/
};

struct fb_videomode {
        const char *name;       /* optional */
        u32 refresh; /*配置帧率，驱动会根据配置参数和帧率计算 pixclock*/
        u32 xres; /*显示有效宽度，按照屏手册填写*/
        u32 yres; /*显示有效高度，按照屏手册填写*/
        u32 pixclock;
        u32 left_margin; /*tft 屏时按照手册填写，当 smart lcd 时赋值 0*/
```

```
        u32 right_margin; /*tft 屏时按照手册填写，当 smart lcd 时赋值 0*/
        u32 upper_margin; /*tft 屏时按照手册填写，当 smart lcd 时赋值 0*/
        u32 lower_margin; /*tft 屏时按照手册填写，当 smart lcd 时赋值 0*/
        u32 hsync_len; /*tft 屏时按照手册填写，当 smart lcd 时赋值 0*/
        u32 vsync_len; /*tft 屏时按照手册填写，当 smart lcd 时赋值 0*/
        u32 sync;  /*当 mipi dsi tft lcd 时需要赋值（FB_SYNC_HOR_HIGH_ACT & FB_SYNC_VERT_HIGH_ACT)，其他的
屏幕不关注*/
        u32 vmode;  /*默认值：FB_VMODE_NONINTERLACED*/
        u32 flag;
};
```

6.3.2.2 smart lcd 配置

```
struct smart_config {
        unsigned int te_switch;    /*smart lcd te 功能控制*/
        unsigned int te_mipi_switch; /*设置 mipi dsi smart lcd te 功能控制*/
        unsigned int te_md;  /*0:te 前沿有效，1:后沿有效*/
        unsigned int te_dp;  /*0:te 低电平有效，1:高电平有效*/
        unsigned int te_anti_jit; /*0:te 信号保持 1 个 pixclk 有效，1:te 信号保持 3 个 pixclk 有效*/
        unsigned int dc_md;  /*0:DC 高电平数据，低电平命令，1:DC 高电平命令，低电平数据*/
        unsigned int wr_md;  /*0:下降沿采样，1:上升沿采样 */
        enum smart_lcd_type smart_type; /*smart slcd 种类，支持 6800/8080/spi-3/spi-4*/
        enum smart_lcd_format pix_fmt;  /*总线数据格式，支持 565,666 等*/
        enum smart_lcd_dwidth dwidth;  /*数据总线宽度*/
        enum smart_lcd_cwidth cwidth;  /*命令总线宽度*/
        unsigned int bus_width;

        unsigned long write_gram_cmd;  /*发送数据前需要发送的命令，默认 0x2c*/
        unsigned int length_cmd; /*不需要实现*/
        struct smart_lcd_data_table *data_table; /*配置屏幕命令表*/
        unsigned int length_data_table; /*配置屏幕命令数量*/
        int (*init) (void); /*不需要实现*/
        int (*gpio_for_slcd) (void); /*不需要实现*/
};
```

6.3.2.3 tft lcd 配置

```
struct tft_config {
        unsigned int pix_clk_inv; /*0:pixclk 默认输出， 1:反转 pixclk*/
        unsigned int de_dl; /*0:DE 引脚高电平输出有效数据， 1:低电平输出有效数据*/
        unsigned int sync_dl; /*0:vsync 和 hsync 引脚高电平输出有效数据， 1:低电平输出有效数据*/
        enum tft_lcd_color_even color_even; /*偶数行时总线 RGB 顺序*/
        enum tft_lcd_color_odd color_odd;  /*奇数行时总线 RGB 顺序*/
```

```
        enum tft_lcd_mode mode; /*总线数据格式，支持 888/666/565*/
};
```

6.3.2.4 mipi dsi lcd 配置

```
struct jzdsi_data jzdsi_pdata = {
        .modes = &panel_modes, /*显示模式*/
        .video_config.no_of_lanes = 2,  /*按照硬件连接填写，支持 1、2lane*/
        .video_config.virtual_channel = 0, /*默认值:0*/
        .video_config.color_coding        =        COLOR_CODE_24BIT,        /*RGB888:COLOR_CODE_24BIT,
RGB565:COLOR_CODE_16BIT_CONFIG1，注：当 smart lcd 时需要配置 COLOR_CODE_24BIT*/
        .video_config.video_mode = VIDEO_BURST_WITH_SYNC_PULSES, /*默认值:VIDEO_BURST_WITH_SYNC_PULSES*/
        .video_config.receive_ack_packets = 0, /*默认值:0*/
        .video_config.is_18_loosely = 0,  /*默认值:0*/
        .video_config.data_en_polarity = 1, /*默认值:1*/
        .video_config.byte_clock = 0, /*默认值:0,驱动根据配置参数自动计算*/
        .video_config.byte_clock_coef = MIPI_PHY_BYTE_CLK_COEF_MUL6_DIV5, /*byte_clock 系数，需要根据实际
情况变动系数，保证正常显示情况下系数越小越好，例：MUL6_DIV5=1.2(乘 6 除 5)*/

        .dsi_config.max_lanes = 2, /*固定值:2*/
        .dsi_config.max_hs_to_lp_cycles = 100, /*默认值:100*/
        .dsi_config.max_lp_to_hs_cycles = 40, /*默认值:40*/
        .dsi_config.max_bta_cycles = 4095, /*默认值:4095*/
        .dsi_config.color_mode_polarity = 1, /*默认值：1*/
        .dsi_config.shut_down_polarity = 1,  /*默认值：1*/
        .dsi_config.max_bps = 2750,     /*默认值:2.75Gbps*/
        .bpp_info = 24, /*RGB888:24 RGB565:16，注：当 smart lcd 时需要配置 24*/
};
```

1. mipi dst tft lcd 配置

当屏幕是 mipi dsi tft lcd 时，赋值 tft_config 需按照下面配置

```
static struct tft_config tft_cfg = {
        .pix_clk_inv = 0, /*固定值：0*/
        .de_dl = 0, /*固定值：0*/
        .sync_dl = 0, /*固定值：0*/
        .color_even = TFT_LCD_COLOR_EVEN_RGB, /*固定值*/
        .color_odd = TFT_LCD_COLOR_ODD_RGB, /*固定值*/
        .mode     =     TFT_LCD_MODE_PARALLEL_888,     /*RGB888    :    TFT_LCD_MODE_PARALLEL_888    ,
RGB565:TFT_LCD_MODE_PARALLEL_565*/
};
```

2. mipi dsi smart lcd 配置

```
当屏幕是 mipi dsi smart lcd 时，配置 smart_confg 时需要按照下面配置 struct smart_config smart_cfg = {
```

```
        .dc_md = 0, /*固定值：0*/
        .wr_md = 1, /*固定值：1*/
        .smart_type = SMART_LCD_TYPE_8080, /*固定值:SMART_LCD_TYPE_8080*/
        .pix_fmt = SMART_LCD_FORMAT_888,  /*固定值:SMART_LCD_FORMAT_888*/
        .dwidth = SMART_LCD_DWIDTH_24_BIT, /*固定值:SMART_LCD_DWIDTH_24_BIT*/
        .te_mipi_switch = 1,  /*1:打开 te 功能，0:关闭 te 功能，使用 PB27 的 func2 做 te 引脚，检测到高电平就
输出数据*/
        .te_switch = 1, /*1:打开 te 功能，0：关闭 te 功能*/
};
```

3. mipi dsi 屏幕寄存器通用配置方式

方式一:

发送大于等于 2 个参数:

```
        struct dsi_cmd_packet cmd = {0x39, 0x05, 0x00, {0x2A, 0x00, 0x00, 0x02, 0xCF}}
```

0x39: 命令种类，发送大于等于 2 个参数

0x05: 参数个数，5 个参数

0x00: 无意义，默认填充 0x00

{0x2A, 0x00, 0x00, 0x02, 0xCF}: 5 个参数值

发送 2 个参数:

```
        struct dsi_cmd_packet cmd = {0x15, 0xC2, 0x08}
```

0x15: 命令种类，发送 2 个参数

0xC2: 第一个参数

0x08: 第二个参数

发送 1 个参数

```
        struct dsi_cmd_packet cmd = {0x05, 0x10, 0x00}
```

0x05: 命令种类，发送 1 个参数

0x10: 第一个参数

0x00: 无意义，默认填充 0x00

6.3.2.5 pwm 背光配置

需要使用 pwm 背光时可进行如下配置:

```
&pwm {
      pinctrl-names="default";
      pinctrl-0=<&pwm0_pd>;     /*按照实际情况配置 pwm 的 gpio*/
      status="okay";
};

backlight {
      compatible="pwm-backlight";
      pwms=<&pwm 1 1000000>; /*选择 pwm1 控制背光，设置 period1000000ns*/
      brightness-levels=<0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15>;/*背光等级，可根据需求调整*/
      default-brightness-level=<4>;
```

```
}
```

6.4 内核编译配置

内核进行控制器和显示屏及背光配置，配置说明如下：

控制器驱动配置（FB_INGENIC_STATE）：

```
Symbol: FB_INGENIC_STAGE [=y]
Type  : tristate
Prompt: Ingenic Framebuffer Driver for stage
  Location:
    -> Device Drivers
      -> Graphics support
        ->Frame buffer Devices
          ->Ingenic Framebuffer Driver (FB_INGENIC [=y])
  Defined at drivers/video/fbdev/ingenic/fb_stage/Kconfig:1
  Depends on: HAS_IOMEM [=y] && FB_INGENIC [=y]
  Selects: FB_INGENIC_DISPLAYS_STAGE[=y] && FB_CFB_FILLRECT [=y] && FB_CFB_COPYAREA [=y] &&
FB_CFB_IMAGEBLIT [=y]
```

pwm 背光驱动配置（PWM_INGENIC_V2）：

```
Symbol: PWM_INGENIC_V2 [=y]
Type  : tristate
Prompt: Ingenic PWM V2 support
    Location:
      -> Device Drivers
        -> Pulse-Width Modulation (PWM) Support (PWM [=y])
    Defined at drivers/pwm/Kconfig:202
    Depends on: PWM [=y] && MACH_XBURST2 [=y]
```

显示屏配置（STAGE_MA0060）

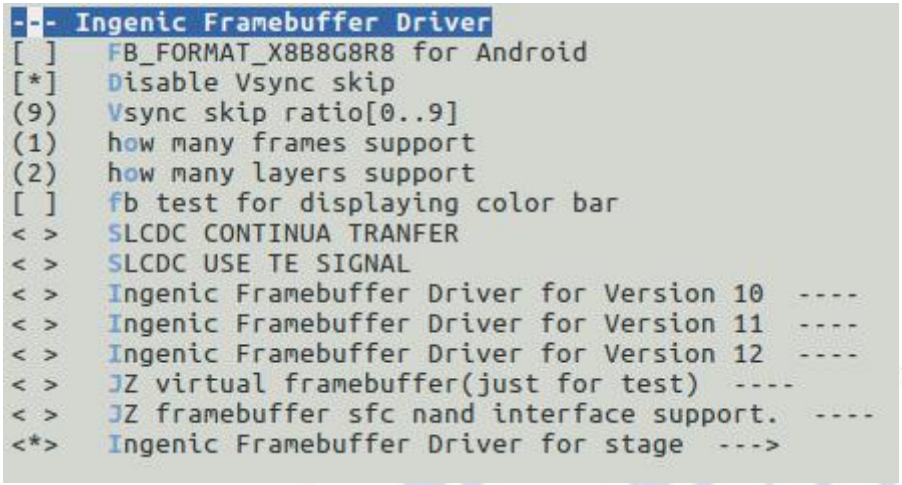
mipi dsi 接口的屏幕支持自动探测，自动探测会挨个判断选中的屏幕是否可用，当有一款屏幕可用或者所有的屏幕都不可用时停止探测。如下为选择屏 ma0060。

```
Symbol: STAGE_MA0060 [=y]
Type  : tristate
Prompt: lcd panel MA0060
    Location:
      -> Device Drivers
        -> Graphics support
          -> Frame buffer Devices
            -> Ingenic Framebuffer Driver (FB_INGENIC [=y])
              -> Supported lcd panels (FB_INGENIC_DISPLAYS_STAGE [=y])
    Defined at drivers/video/fbdev/ingenic/fb_stage/displays/Kconfig:21
```

Depends on: HAS_IOMEM [=y] && FB_INGENIC [=y] && FB_INGENIC_DISPLAYS_STAGE [=y]

6.4.1 内核默认编译配置

内核默认配置 DPU 驱动，MA0060 和 FW050 显示屏自动探测，配置界面如下：



6.4.2 内核自定义编译配置

用户可根据实际需求去掉该驱动的配置，或配置其他显示屏的驱动。

6.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/fb0
/dev/fb1
/dev/fb2
/dev/fb3
```

6.6 注意事项

最多支持 2 层缩放，缩放可以是任意 2 层

只有 1,2 层支持 yuv422、nv12、nv21

6.7 调试屏幕列表

屏幕	种类	分辨率
panel-ma0060	mipi slcd	720*1280
panel-jd9161z	mipi tft	480*800

panel-st7701s	mipi tft	480*800
panel-kd050hdfia019	mipi tft	480*854
panel-tl040hds01ct	mipi tft	720*720
panel-ylym286a	mipi tft	1920*1080
panel-tl040wvs03ct	tft	480*480
panel-y88249	tft	640*480
panel-yts500xlai	tft	720*1280
panel-kd035hvfbd037	slcd	320*480
panel-kd035hvfmd057	slcd	320*480
panel-gc9203	slcd	220*128
panel-kd050wvfpa029	tft	800*400
panel-fw050	mipi tft	720*1280

7 Rotator 图像旋转

7.1 模块功能介绍

介绍模块的基本功能，驱动实现的功能，驱动实现的架构和框架图。默认配置的功能，需要自己配置的功能等内容

输入格式支持：RGB888 ARGB88 RGB565 RGB555 ARGB1555 YUV422

输出格式支持： ARGB8888，RGB565，RGB555，YUV422

旋转角度：0°，90°，180°，270°，Horizontal mirror，Vertical mirror

不支持 RGB 和 YUV 格式互相转换，支持 RGB 格式互相转换。

7.2 驱动源码位置

驱动源码所在位置：

```
drivers/media/platform/ingenic-rotate/
```

7.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dts
```

ROTATE 控制器描述：

```
rotate : rotate@0x13070000 {
        compatible="ingenic,x2000-rotate";
        reg=<0x130700000 x10000>;
        interrupt-parent=<&core_intc>;
        interrupts=<IRQ_ROTATE>;
        status="okay";
};
```

7.3.1 设备树默认配置

设备树默认编译会产生 rotate 设备。

7.3.2 设备树自定义配置

用户可根据实际需求关闭 rotate 设备，将该节点配置为 disabled：

```
&rotate  {
        status="disabled";
};
```

7.4 内核编译配置

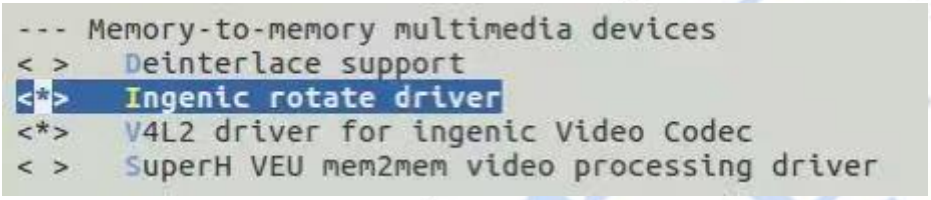
内核配置 VIDEO_INGENIC_ROTATE，配置说明如下：

```
Symbol : VIDEO_INGENIC_ROTATE[=y]
Type : tristate
Prompt : Ingenic rotate driver
```

```
Location :
  ->Device Drivers
    ->Multimedia support \ (MEDIA\_SUPPORT\[=y\]\)
      ->Memory-to-memory multimedia devices \(V4L\_MEM2MEM\_DRIVERS\[=y\]\)
Defined at drivers/media/platform/Kconfig:174
Depends on : MEDIA_SUPPORT [=y] && V4L_MEM2MEM_DRIVERS [=y] && VIDEO_DEV [=y] && VIDEO_V4L2 [=y] &&
(SOC_X2000 [=n] || SOC_X2000_V12 [=y] || SOC_M300 [=n] )
Selects : VIDEOBUF2_DMA_CONTIG [=y] && V4L2_MEM2MEM_DEV [=y]
```

7.4.1 内核默认编译配置

内核默认配置 rotate 驱动，配置界面如下：



7.4.2 内核自定义编译配置

用户可根据实际需求去掉该驱动的配置。

7.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/video0
```

rotate 的设备节点

7.6 应用程序使用说明

7.6.1 rotate

7.6.1.1 源码位置

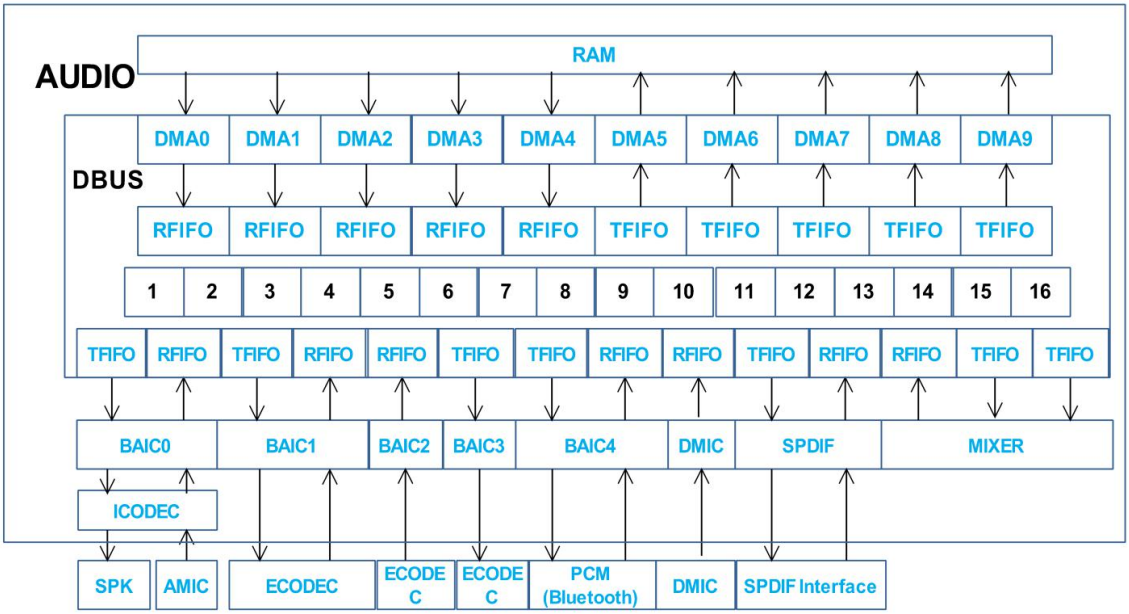
```
packages/example/Sample/rotate/
```

7.6.1.2 测试方法

测试应用在 testsuite/rotate，测试了在不同分辨率下进行水平镜像，垂直镜像，旋转 0 度，旋转 90 度，旋转 180 度，旋转 270 度。

8 Audio 音频子系统

8.1 模块功能介绍



图表 8- 1 音频子系统硬件模块

1. audio 有 10 个 DMA 模块，其中 DMA0-DMA4 功能是从 RAM 搬移数据到 fifo，DMA5-DMA9 功能是从 fifo 搬数据到 RAM 中，每个 DMA 固定连接到 DBUS 中的一个 FIFO 上。

2. 5 个 BAIC（Basic Audio Interface Controller），BAIC0 和 ICODEC(Internal CODEC)相连支持单通道录放音。

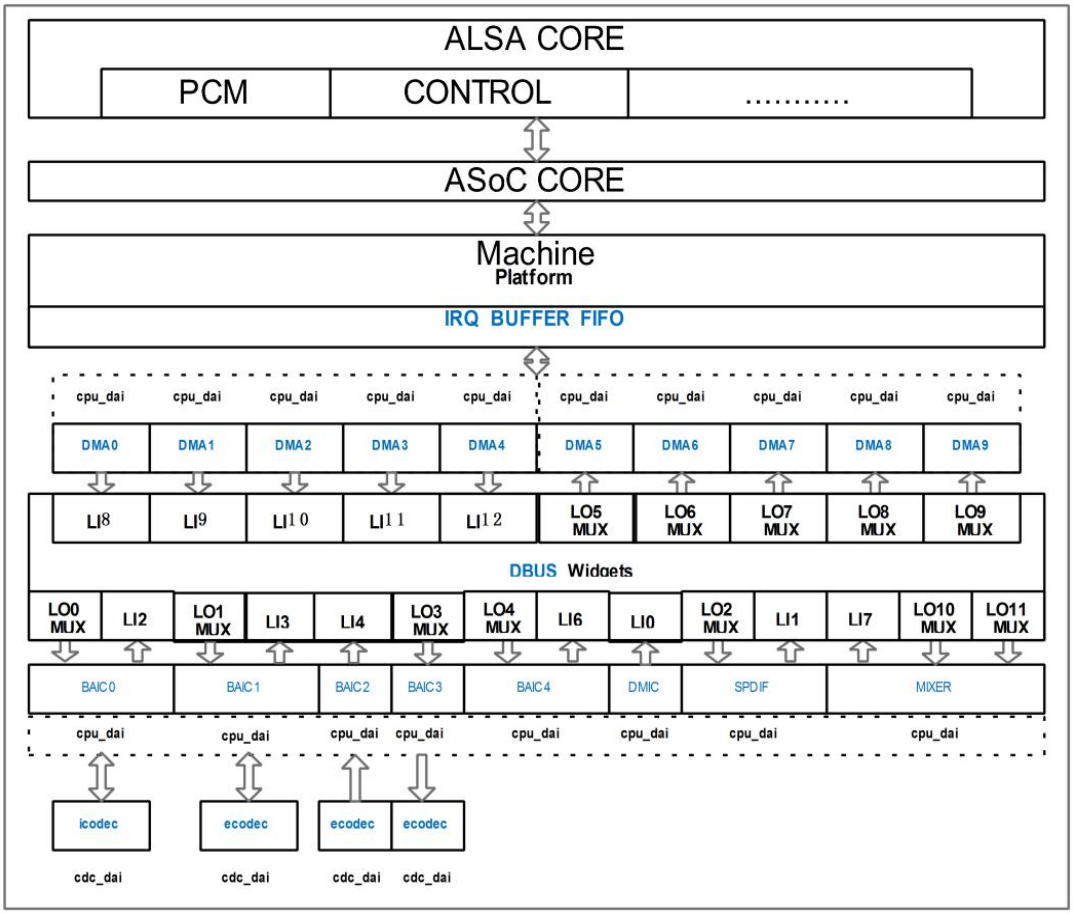
3. dmic 最大支持 8 通道，支持语音唤醒。

4. spdif（Sony Philips Digital Interconnect Format）。

5. DBUS 可以把 RFIFO 和 TFIFO 动态连接在一起形成通路，所以通过 DBUS 的配置可以把 DMA 和其他子模块连接成数据流通路。

模块名	支持协议	支持通道数	功能
BAIC0+ICODEC		1 通道（限制于 ICODEC）	录音/放音
BAIC1	I2S/LEFT/RIGHT/PCMA/PCMB/DSPA/DSPB	1/2 通道	录音/放音
BAIC2	TDM1A/TDM1B/TDM2A/TDM2B/I2S7+1/LEFT7+1/RIGHT7+1	2/4/8 通道	录音
BAIC3	TDM1A/TDM1B/TDM2A/TDM2B/I2S7+1/LEFT7+1/RIGHT7+1	2/4/8 通道	放音
BAIC4	I2S/LEFT/RIGHT/PCMA/PCMB/DSPA/DSPB	1/2 通道	录音/放音
DMIC	PulseDensityModulation	1/2/4/6/8 通道	录音
SPDIF	BMC(BiphaseMarkCode)	2 通道	录音/放音

8.2 软硬件模块对应关系



图表 8- 2 音频子系统软硬件模块对应关系

amixer 配置数据通路时的软硬件对应关系：

硬件名称	软件名称
BAIC0_PLAYBACK	LO0_MUX
BAIC1_PLAYBACK	LO1_MUX
BAIC2_PLAYBACK	LO2_MUX
BAIC3_PLAYBACK	LO3_MUX
BAIC4_PLAYBACK	LO4_MUX
DMA5	LO5_MUX
DMA6	LO6_MUX
DMA7	LO7_MUX
DMA8	LO8_MUX
DMA9	LO9_MUX
MIX_IN0	LO10_MUX
MIX_IN1	LO11_MUX

数据通路输入端软硬件对应关系：

硬件名称	软件名称
DMIC_CAPTURE	LI0
SPDIF_CAPTURE	LI1
BAIC0_CAPTURE	LI2
BAIC1_CAPTURE	LI3
BAIC2_CAPTURE	LI4
BAIC3_CAPTURE	LI5
BAIC4_CAPTURE	LI6
MIX0_OUT	LI7
DMA0	LI8
DMA1	LI9
DMA2	LI10
DMA3	LI11
DMA4	L12

8.3 驱动源码位置

audio 控制器源码位置：

```
sound/soc/ingenic/as-v2
```

板级源码位置：

```
sound/soc/ingenic/board
```

内部 codec 源码位置：

```
sound/soc/ingenic/icodec/icdc_inno.c
```

外部 codec 源码位置：

```
sound/soc/ingenic/ecodec/
```

8.4 设备树配置

设备树所在位置

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

设备树描述：

```
as:as {
        compatible = "simple-bus";
        #address-cells = <1>;
        #size-cells = <1>;
        ranges = <>;

        as_platform: as-platform {
                compatible = "ingenic,as-platform";
                reg = <0x134d0000 0x114>, <0x134d1000 0x100>;
```

```
                reg-names = "dma", "fifo";
                ingenic,fifo-size = <4096>;
                interrupt-parent = <&core_intc>;
                interrupts = <IRQ_AUDIO>;
                ingenic,fth_quirk;
        };
        as_virtual_fe: as-virtual-fe {
                compatible = "ingenic,as-vir-fe";
                reg = <0x00000000 0x0>;
                ingenic,cap-dai-bm = <0xc>;
                ingenic,num-dais = <4>;
        };
        as_fmtcov: as-fmtcov {
                compatible = "ingenic,as-fmtcov";
                reg = <0x134d2000 0x28>;
        };
        as_fe_dsp: as-dsp {
                compatible = "ingenic,as-dsp";
                reg = <0x134d4000 0x30>;
                ingenic,li-port = <0 1 2 3 4 6 7 8 9 10 11 12>;
                ingenic,lo-port = <0 1 2 3 4 5 6 7 8 9 10 11>;
                ingenic,cap-dai-bm = <0x3e0>;
                ingenic,num-dais = <10>;
        };
        as_be_baic: as-baic {
                compatible = "ingenic,as-baic";
                reg = <0x134d5000 0x5000>;
                ingenic,num-dais = <5>;
                /* using dai-array to determine which BAIC to use */
                ingenic,dai-array = <0>, <1>, <2>, <3>, <4>;
                ingenic,dai-mode = <BAIC_3AND(BAIC_PCM_MODE, BAIC_DSP_MODE, BAIC_I2S_MODE)>,
                                <BAIC_3AND(BAIC_PCM_MODE, BAIC_DSP_MODE, BAIC_I2S_MODE)>,
                                <BAIC_4AND(BAIC_I2S_MODE, BAIC_TDM1_MODE, BAIC_TDM2_MODE, BAIC_NO_REPLAY)>,
                                <BAIC_4AND(BAIC_I2S_MODE, BAIC_TDM1_MODE, BAIC_TDM2_MODE, BAIC_NO_RECORD)>,
                                <BAIC_3AND(BAIC_PCM_MODE, BAIC_DSP_MODE, BAIC_I2S_MODE)>;
                ingenic,data-pin-num = <1>, <1>, <4>, <4>, <1>;
                ingenic,clk-split = <1>, <1>, <1>, <1>, <0>;
                ingenic,clk-rname = "div_i2s0","div_i2s0","div_i2s2","no_clk","mux_pcm";
                ingenic,clk-tname = "div_i2s1","div_i2s1","no_clk","div_i2s3","mux_pcm";
                ingenic,pcm-clk-parent = "div_i2s2";
        };
as_dmic: as-dmic {
                compatible = "ingenic,as-dmic";
```

```
                reg = <0x134da000 0x10>;
                ingenic,clk-name = "mux_dmic"
                ingenic,clk-parent = "ext";
        };
        as_aux_mixer: as-mixer {
                compatible = "ingenic,as-mixer";
                reg = <0x134dc000 0x8>;
                ingenic,num-mixers = <1>;
        };
        as_spdif: as-spdif {
                compatible = "ingenic,as-spdif";
                reg = <0x134db000 0x14>, <0x134db100 0x14>;
                reg-names = "out", "in";
                ingenic,clk-name = "mux_spdif";
                ingenic,clk-parent = "div_i2s2";
         };
};
```

8.4.1 设备树默认配置

设备树默认编译会产生 audio 设备，并在 arch/mips/boot/dts/ingenic/halley5_v20.dts 下配置：

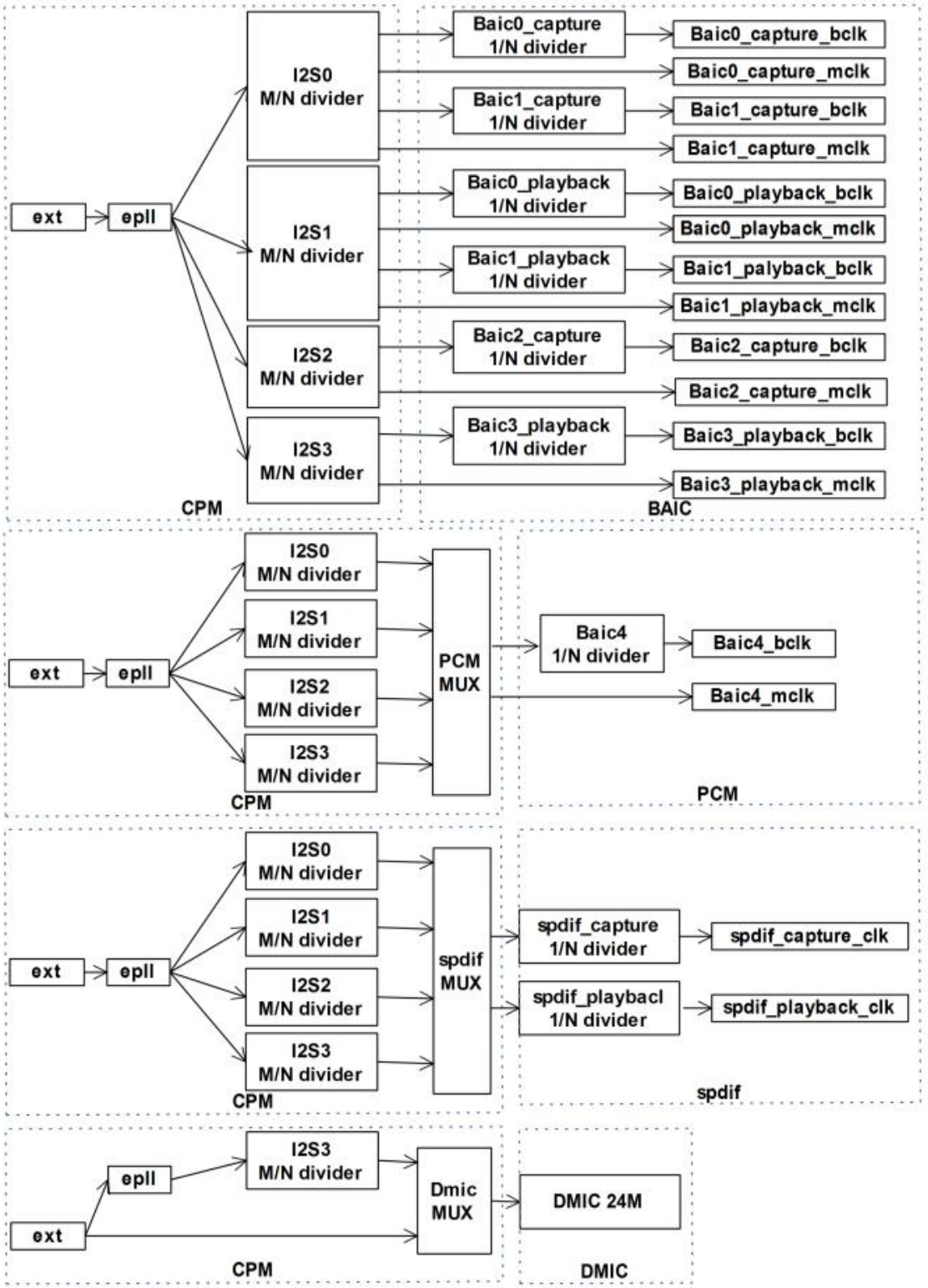
```
&as_be_baic {
        pinctrl-names = "default";
        pinctrl-0 =  <&baic4_pd>;
};

&as_dmic {
        pinctrl-names = "default";
        pinctrl-0 = <&dmic_pc_4ch>;
};

&icodec {
        ingenic,spken-gpio = <&gpb 2 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
};
```

```
sound {
                compatible = "ingenic,x2000-sound";
                ingenic,model = "halley5_v20";
};
```

8.4.2 设备树自定义配置

用户可根据实际需求关闭 audio 设备,或配置 ingenic，spken-gpio 属性。

8.4.3 时钟树配置



时钟树配置注意事项：

1. 当 baic0 和 baic1 同时录音或同时放音时，两设备必须配置相同的采样率，位宽，通道数。原因是 baic0 连接的 icodec 要求供应的工作时钟必须是采样率的 256 倍，所以 baic1 只能跟着 baic0.

2. spdif 选择父时钟尽量选择一个空闲时钟，如果和其他设备共享父时钟需要设置一个都能满足的

频率。

8.5 内核编译配置

内核配置 SND_ASOC_INGENIC，配置说明如下：

```
Symbol: SND_ASOC_INGENIC [=y]
Type  : tristate
Prompt: ASoC support for Ingenic
  Location:
    -> Device Drivers
      -> Sound card support (SOUND [=y])
        -> Advanced Linux Sound Architecture (SND [=y])
          -> ALSA for SoC audio support (SND_SOC [=y])
  Defined at sound/soc/ingenic/Kconfig:1
  Depends on: SOUND [=y] && !M68K && !UML && SND [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y]) && SND_SOC
[=y]

/*AudioSystem Version 2 For Ingenic SOCs  支持 X2000_V12 和 M300*/
Prompt: Audio Version:
  Location:
    -> Device Drivers
      -> Sound card support (SOUND [=y])
        -> Advanced Linux Sound Architecture (SND [=y])
          -> ALSA for SoC audio support (SND_SOC [=y])
            -> ASoC support for Ingenic (SND_ASOC_INGENIC [=y])
  Defined at sound/soc/ingenic/Kconfig:15
  Depends on: SOUND [=y] && !M68K && !UML && SND [=y] && SND_SOC [=y] && SND_ASOC_INGENIC [=y]
  Selected by: SOUND [=y] && !M68K && !UML && SND [=y] && SND_SOC [=y] && SND_ASOC_INGENIC [=y] && m

/* 选择板级，以 x2000_v12 为例  */
Prompt: SOC x2000 v12 codec Type select
  Location:
    -> Device Drivers
      -> Sound card support (SOUND [=y])
        -> Advanced Linux Sound Architecture (SND [=y])
          -> ALSA for SoC audio support (SND_SOC [=y])
            -> ASoC support for Ingenic (SND_ASOC_INGENIC [=y])
              -> Ingenic Board Type Select
  Defined at sound/soc/ingenic/Kconfig:125
  Depends on: SOUND [=y] && !M68K && !UML && SND [=y] && SND_SOC [=y] && SND_ASOC_INGENIC [=y] && SOC_X2000_V12
[=y] && \
SND_ASOC_INGENIC_AS_V2 [=y]
  Selected by: SOUND [=y] && !M68K && !UML && SND [=y] && SND_SOC [=y] && SND_ASOC_INGENIC [=y] && SOC_X2000_V12
[=y] && \
```

```
SND_ASOC_INGENIC_AS_V2 [=y] && m
```

8.5.1 内核默认编译配置

内核默认配置 audio 驱动，配置界面如下：

```
--- ALSA for SoC audio support
< >   SoC Audio for the Atmel System-on-Chip
< >   Synopsys I2S Device Driver
      SoC Audio for Freescale CPUs  --->
      Allwinner SoC Audio support  --->
< >   XTFPGA I2S master
<*>   ASoC support for Ingenic  --->
      CODEC drivers  --->
< >   ASoC Simple sound card support
```

8.5.2 内核自定义编译配置

用户可根据实际需求去掉该驱动的配置。

8.6 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/snd/controlC0
/dev/snd/pcmC0D0p
/dev/snd/pcmC0D1p
/dev/snd/pcmC0D2p
/dev/snd/pcmC0D3p
/dev/snd/pcmC0D4p
/dev/snd/pcmC0D5c
/dev/snd/pcmC0D6c
/dev/snd/pcmC0D7c
/dev/snd/pcmC0D8c
/dev/snd/pcmC0D9c
/dev/snd/timer
```

10 个 pcmC0D*设备对应硬件的 10 个 DMA 设备

8.7 应用程序使用说明

8.7.1 baic0+icodec 录放音

8.7.1.1 baic0+icodec 录音

amixer 命令配置 BAIC0_CAPTURE(LI2)和 DM6(LO6_MUX)数据通路连接,arecord 命令录音

```
amixer cset name='LO6_MUX' LI2
arecord -D hw:0,6 -c 1 -f S32_LE -r 16000 -d 5 baic0_16000-32-1.wav
arecord -D hw:0,6 -c 1 -f S16_LE -r 48000 -d 5 baic0_48000-16-1.wav
```

8.7.1.2 baic0+icodec 放音

amixer 命令配置 BAIC0_PLAYBACK(LO0_MUX)和 DMA0(LI8)数据通路连接,aplay 命令放音

```
amixer cset name='LO0_MUX' LI8
aplay -D hw:0,0 16000-32-1.wav
```

8.7.1.3 调整 icodec 录放音增益

```
amixer cset name='ICODEC MICL GAIN' 31
amixer cset name='ICODEC HPOUTL GAIN' 31
```

8.7.2 baic1 录放音

8.7.2.1 baic1 录音

amixer 命令配置 BAIC1_CAPTURE(LI3)和 DMA7(LO7_MUX)数据通路连接,arecord 命令录音

```
amixer cset name='LO7_MUX' LI3
arecord -D hw:0,7 -c 2 -f S32_LE -r 48000 -d 10 baic1_48000-32-2.wav
arecord -D hw:0,7 -c 2 -f S16_LE -r 16000 -d 10 baic1_16000-16-2.wav
```

8.7.2.2 baic1 放音

amixer 命令配置 BAIC1_PLAYBACK(LO1_MUX)和 DMA1(LI9)数据通路连接,aplay 命令放音

```
amixer cset name='LO1_MUX' LI9
aplay -D hw:0,1 48000-16-2.wav
```

8.7.3 baic2 录音

amixer 命令配置 BAIC2_CAPTURE(LI4)和 DMA8(LO8_MUX)数据通路连接,arecord 命令录音

```
amixer cset name='LO8_MUX' LI4
arecord -D hw:0,8 -r 48000 -f S32_LE -c 8 -d 8 baic2_48000-32-8.wav
```

8.7.4 baic3 放音

amixer 命令配置 BAIC3_PLAYBACK(LO3_MUX)和 DMA2(LI10)数据通路连接,aplay 命令放音

```
amixer cset name='LO3_MUX' LI10
aplay -D hw:0,2 48000-32-8.wav
```

8.7.5 baic4 录放音

略

8.7.6 dmic 录音

8.7.6.1 dmic 录音

amixer 命令配置 DMIC_CAPTURE(LI0)和 DMA5(LO5_MUX)数据通路连接，**dmic 只能和 DMA5 连接成数据通路**，arecord 命令录音

```
amixer cset name=LO5_MUX LI0
arecord -D hw:0,5 -c 2 -f S16_LE -r 16000 -d 10 dmic_16000-16-2.wav
```

```
arecord -D hw:0,5 -c 4 -f S16_LE -r 48000 -d 10 dmic_48000-16-4.wav
arecord -D hw:0,5 -c 6 -f S16_LE -r 16000 -d 10 dmic_16000-16-6.wav
arecord -D hw:0,5 -c 8 -f S16_LE -r 16000 -d 10 dmic_16000-16-8.wav
```

8.7.6.2 调整 dmic 录音增益

```
amixer cset name='DMIC Volume' 31
```

8.7.7 spdif 录放音

8.7.7.1 spdif 放音

amixer 命令配置 SPDIF_PLAYBACK(LO2_MUX)和 DMA4(LI12)数据通路连接,aplay 命令放音

```
amixer cset name='LO2_MUX' LI12
aplay -D hw:0,4 48000-16-2.wav
```

8.7.7.2 spdif 录音

amixer 命令配置 SPDIF_CAPTURE(LI1)和 DMA9(LO8_MUX)数据通路连接,arecord 命令录音

```
amixer cset name='LO9_MUX' LI1
arecord -D hw:0,9 -c 2 -f S16_LE -r 48000 -d 10 spdif_48000-16-2.wav
```

8.7.8 使用注意事项

1. 录音位宽为 24bit 时，录音数据在内存用 32bit 存放，有效数据存放在低 24bit，使用上位机播放时需要预处理，如果使用 x2000/m300 audio 设备播放不需要预处理。

2. spdif 录音数据由音频数据和音频信息组成，播放前需要预处理，spdif 具体格式请阅读 PM 手册。

9 DDR 控制器接口

10NEMC 外部存储控制器接口

11SPI Flash 控制器接口

11.1 模块功能介绍

SFC 是 spi master 设备，用来控制 spi flash。支持 CPU 模式和 DMA 模式传输。

- 支持 CDT 命令传输模式。
- 支持 DMA descriptor chain 数据传输模式。
- 支持硬件 polling flash 的状态。
- 支持 Standard, Dual, Quad, Octal SPI 等多种传输协议。

11.2 驱动源码位置

驱动源码所在位置：

```
kernel/drivers/mtd/devices/ingenic_sfc_v2/
├── ingenic_sfc_common.c
├── ingenic_sfc_nand.c
├── ingenic_sfc_nor.c
├── ingenic_sfc_ops.c
└── nand_device
    ├── ato_nand.c
    ├── dosilicon_nand.c
    ├── foresee_nand.c
    ├── gd_nand.c
    ├── mxic_nand.c
    ├── nand_common.c
    ├── xtx_mid0b_nand.c
    ├── xtx_nand.c
    └── zetta_nand.c
```

11.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

设备树描述：

```
sfc:sfc@0x13440000 {
    compatible = “ingenic,sfc”;
    reg = <0x13440000 0x10000>;
    interrupt-parent = <&core_intc>;
    interrupts = <IRQ_SFC>;
};
```

11.3.1 设备树默认配置

设备树默认编译会产生 sfc 设备

```
&sfc {
```

```
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&sfc_pe>;
        ingenic,sfc-max-frequency = <400000000>;
        ingenic,use_ofpart_info  = /bits/ 8 <0>;
        ingenic,spiflash_param_offset = <0>;

        norflash@0 {
                partitions {
                        compatible = "fixed-partitions";
                        #address-cells = <1>;
                        #size-cells = <1>;

                        /* spi nor flash partition */
                        uboot@0 {
                                label = "uboot";
                                reg = <0x0000000 0x40000>;
                                /*read-only;*/
                        };

                        kernel@40000 {
                                label = "kernel";
                                reg = <0x40000 0x300000>;
                        };

                        rootfs@360000 {
                                label = "rootfs";
                                reg = <0x360000 0xca0000>;
                        };
                };
        };

        nandflash@1 {
                partitions {
                        compatible = "fixed-partitions";
                        #address-cells = <1>;
                        #size-cells = <1>;

                        /* spi nand flash partition */
                        partition@0 {
                                label = "uboot";
                                reg = <0x0000000 0x100000>;
                                /*read-only;*/
```

```
                    };

                    partition@100000 {
                            label = "kernel";
                            reg = <0x100000 0x800000>;
                    };

                    partition@900000 {
                            label = "rootfs";
                            reg = <0x900000 0xf700000>;
                    };
            };
    };
};
```

11.3.2 设备树自定义配置

用户可根据实际需求关闭该节点，或进行以下配置：

属性名称	说明
● ingenic,sfc-max-frequency	配置 sfc 最大频率,需要经过 4 分频,实际线上时钟为(*sfc-max-frequency / 4) MHz*
● ingenic,use_ofpart_info	当使用设备树分区信息时,配置为 1.
● ingenic,spiflash_param_offset	当使用 spi flash 存放 flash 参数和分区信息时，用于修改 spi flash 中存放参数的偏移地址。当值为 0 时,默认偏移为 0x5800。

11.4 内核编译配置

内核配置 INGENIC_SFC，配置说明如下：

```
Symbol: INGENIC_SFC [=y]
Type  : tristate
Prompt: Ingenic series SFC driver
  Location:
    -> Device Drivers
      -> Memory Technology Device (MTD) support (MTD [=y])
        -> Self-contained MTD device drivers
  Defined at drivers/mtd/devices/Kconfig:236
  Depends on: MTD [=y] && HAS_IOMEM [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y])
```

11.4.1 内核默认编译配置

内核默认配置 SFC 驱动，配置界面如下

```
< > Uncached system RAM
< > Physical system RAM
< > Test driver using RAM
< > MTD using block device
    *** Disk-On-Chip Device Drivers ***
< > M-Systems Disk-On-Chip G3
<*> Ingenic series SFC driver
      Select Ingenic series SFC driver version (Use ingenic sfc driver version 2)  --->
      the SFC external memory (nor or nand) (Support ingenic sfc-nand)  --->
```

11.4.2 内核自定义编译配置

用户可根据实际需求去掉该选项的配置。

11.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/mtd*
```

mtd 字符设备节点

```
/dev/mtdblock*
```

块设备节点

11.6 应用程序使用说明

11.6.1 flash 读写测试

1. 执行脚本：StorageMedia-Test.sh

```
#./StorageMedia-Test.sh
*********************************************
* * Please input which module you want test .
*********************************************
* Enter 1 : Read/Write test in the nand/nor .
* Enter 2 : Read/Write test in the sdcard .
* Enter 3 : Read/Write test in the ddr .
* Enter 4 : Read/Write test between nand/nor and sdcard .
* Enter 5 : Read/Write test between nand/nor and ddr .
* Enter 6 : Read/Write test between ddr and sdcard .
```

2. 输入 1

```
You select 1, nand/nor <-> nand/nor
*************************************************
* * Please choice which Size you want copy .
*************************************************
* Enter 1 : auto random Size
* Enter 2 : setting a fix Size
```

3. 输入 1 或输入 2

选择 1:测试随机大小的数据(根据 flash 剩余空间,自己计算大小)

选择 2:测试指定大小的数据(Kbyte)

4. flash 读写测试开始,连续测试一段时间,如果无 error 异常退出则正常。

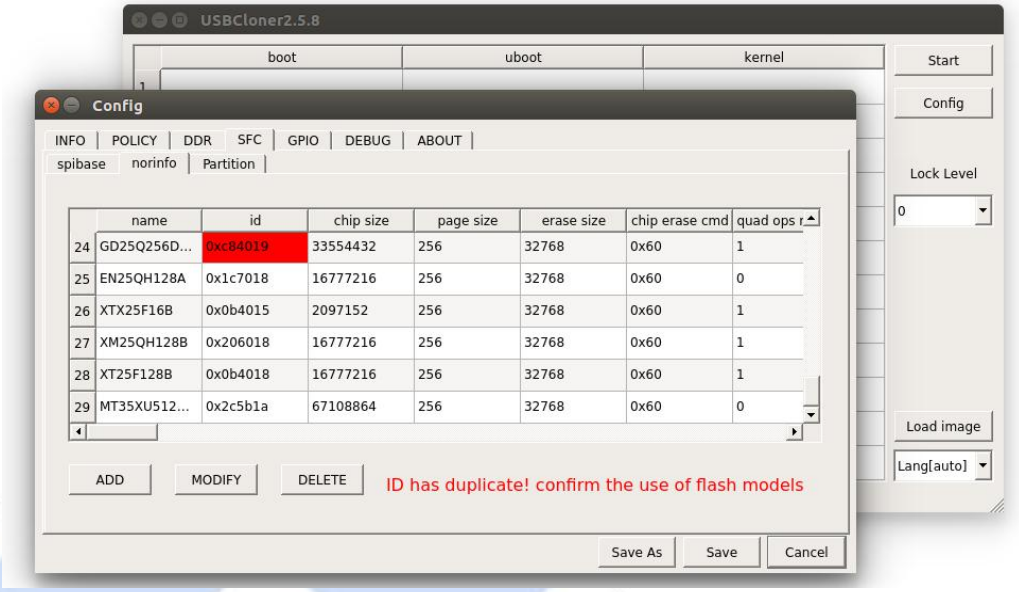
11.6.2 添加一款新的 flash 参数支持（依赖 cloner 烧录工具）

11.6.2.1 添加 spi nor flash 参数

通过 cloner 烧录工具添加 spi nor flash 参数:

1. 打开 cloner 中的 *Config*,在 INFO 菜单下,选择 Board 为“x2000_v12_sfc_nor_lpddr3_linux.cfg”。
2. 在 SFC 菜单下，选择二级菜单 norinfo，选择 ADD。
3. 在弹出的对话框中按照 spi nor flash 的参数填上保存即可。

注：烧录工具中 id 号为红色时，表示存在多个相同 ID 的 flash 参数，需要手动删除冲突的 flash 参数



11.6.2.2 添加 spi nand flash 参数

11.6.2.2.1 添加 nand flash 厂商支持

各厂商 nand flash 参数配置文件位置

```
kernel/drivers/mtd/devices/ingenic_sfc_v2/nand_device/
```

如果存在该厂商的配置文件，则跳过此步骤；如果不存在，则需要参照其他厂商的参数配置文件，编写自己的配置文件。例如:

```
cp gd_nand.c myflash_nand.c; vi myflash_nand.c
```

flash 参数配置文件主要构成（例如 gd_nand.c）

```
/* 1. flash支持型号的个数 */
define GD_DEVICES_NUM 7
```

```
/* 2.  flash 的基础参数  */
static struct ingenic_sfcnand_base_param gd_param[GD_DEVICES_NUM] = {
[0] = {...},
}

/* 3. flash 的 id 和参数建立关联 */
static struct device_id_struct device_id[GD_DEVICES_NUM] = {
DEVICE_ID_STRUCT(0xD1, "GD5F1GQ4UB",&gd_param[0]),
...
}

/* 4. flash 获取默认的 cdt 参数，并根据 id 更新参数*/
static cdt_params_t *gd_get_cdt_params(struct sfc_flash *flash, uint8_t device_id){
...
}

/* 5. 使用通用的 get feature 接口，需要定义处理 ecc 的接口  */
static inline int deal_ecc_status(struct sfc_flash *flash, uint8_t device_id, uint8_t ecc_status){
...
}

/* 6.注册 flash 参数  */
static int __init gd_nand_init(void) {
...
}
```

11.6.2.2.2 添加 nand flash 型号支持

需要对照 flsah 手册，完成以下步骤

1. 添加 flash 基础参数.

```
[6] = {.pagesize = 2 * 1024, ... },
```

2. 建立 flash 与参数的关联

```
DEVICE_ID_STRUCT(0xA1, "GD5F1GQ4RF",&gd_param[6]),
```

3. 根据 id,更新默认参数

```
case 0xA1:
      gd_nand->cdt_params.standard_r.addr_nbyte = 3;
      gd_nand->cdt_params.quad_r.addr_nbyte = 3;
break;
```

4. 在处理 ecc 的接口中添加对应型号的处理

```
switch(device_id) {
```

```
    case 0xA1:
    ...
}
```

5. 将支持 flash 总数＋1

```
#define GD_DEVICES_NUM 7
```

11.6.2.2.3 对照 flash 手册比较默认参数，更新参数

默认 command 参数文件:

```
u-boot/drivers/mtd/devices/jz_sfc_v2/nand_device/nand_common.h
```

制作参数接口说明:

控制 flash 相关:

```
/*CMD_INFO(_CMD, flash 命令, dummy bits, 地址长度, 传输协议)*/
  读 flash 状态的参数
/*ST_INFO(_ST, flash 命令, status 偏移 bit, status 状态位的 mask, status 状态位的期望值, 传输长度, dummy bits)*/
```

例如:

Command Name	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte N
Read From Cache x 4	6BH	dummy (2)	A15-A8	A7-A0	dummy (2)	(D7-D0)x4

```
/*dummy 8bit, 地址长度 3，传输协议对应 sfc 手册:TM_QI_QO_SPI*/
CMD_INFO(cdt_params.quad_r, SPINAND_CMD_RDCH_X4, 8, 2, TM_QI_QO_SPI);
```

Register	Addr	7	6	5	4	3	2	1	0
Status	C0H	Reserved	ECCS2	ECCS1	ECCS0	P_FAIL	E_FAIL	WEL	OIP

```
/*OIP 便宜 bit 是 0，status 状态位 mask 是 0x1, 期望值 0x0，传输长度 1 byte, get feature dummy 0bit*/
ST_INFO(cdt_params.oip, SPINAND_CMD_GET_FEATURE, 0, 0x1, 0x0, 1, 0);
```

默认 command 参数如下

```
/*
* cdt params
*/
#define CDT_PARAMS_INIT(cdt_params) { \
/* read to cache */ \
CMD_INFO(cdt_params.r_to_cache, SPINAND_CMD_PARD, 0, 3, TM_STD_SPI); \
/* standard read from cache */ \
CMD_INFO(cdt_params.standard_r, SPINAND_CMD_FRCH, 8, 2, TM_STD_SPI); \
/* quad read from cache*/ \
CMD_INFO(cdt_params.quad_r, SPINAND_CMD_RDCH_X4, 8, 2, TM_QI_QO_SPI); \
```

```
/* standard write to cache*/ \
CMD_INFO(cdt_params.standard_w_cache, SPINAND_CMD_PRO_LOAD, 0, 2, TM_STD_SPI); \
/* quad write to cache*/ \
CMD_INFO(cdt_params.quad_w_cache, SPINAND_CMD_PRO_LOAD_X4, 0, 2, TM_QI_QO_SPI); \
/* write exec */ \
CMD_INFO(cdt_params.w_exec, SPINAND_CMD_PRO_EN, 0, 3, TM_STD_SPI); \
/* block erase */ \
CMD_INFO(cdt_params.b_erase, SPINAND_CMD_ERASE_128K, 0, 3, TM_STD_SPI); \
/* write enable */ \
CMD_INFO(cdt_params.w_en, SPINAND_CMD_WREN, 0, 0, TM_STD_SPI); \
\
/* get frature wait oip not busy */ \
ST_INFO(cdt_params.oip, SPINAND_CMD_GET_FEATURE, 0, 0x1, 0x0, 1, 0); \
}
```

参数更新(例如 gd_nand.c)

```
static cdt_params_t *gd_get_cdt_params(struct sfc_flash *flash, uint8_t device_id) {
switch(device_id) {
...
case 0xA1:
gd_nand->cdt_params.standard_r.addr_nbyte = 3; /*地址长度更新为 3byte*/
gd_nand->cdt_params.quad_r.addr_nbyte = 3;/*地址长度更新为 3byte*/
break;
...
}
```

11.6.2.2.4 默认 get_feature 接口以及如何实现自己的接口

默认 get feature 接口位置:

```
u-boot/drivers/mtd/devices/jz_sfc_v2/nand_device/nand_common.c
```

使用默认的 get feature 接口需要定义自己的处理 ecc 的接口

```
static inline int deal_ecc_status(struct sfc_flash *flash, uint8_t device_id, uint8_t ecc_status) {
        switch(device_id) {
                case 0xA1:
                case 0xB1 ... 0xB4:
                        switch((ecc_status >> 4) & 0x7) {
                                case 0x7:
                                        ret = -EBADMSG;
                                        break;
                                case 0x6:
                                        ret = 0x8;
                                        break;
                                case 0x5:
```

```
                                ret = 0x7;
                                break;
                        default:
                                ret = 0;
        }
        ...
        break;
}
```

特殊情况需要自己定义 get feature 接口：

1. 在 flash 参数的配置文件中定义自己的 get__feature 接口
2. 将自己定义的接口注册到参数中

```
int32_t my_get_feature(struct sfc_flash *flash, uint8_t flag) {
        ...
}

static int __init gd_nand_init(void) {
        ...
        /* use private get feature interface, please define it in this document */
        //gd_nand->ops.get_feature = NULL;
        gd_nand->ops.get_feature = my_get_feature();
        ...
}
```

11.6.3 内置 flash 参数实现（不使用 cloner 烧录工具）

11.6.3.1 kernel 使用 u-boot 内置参数（依赖 u-boot）

依赖 u-boot.（参考 uboot 文档）

kernel 使用默认配置即可.

11.6.3.2 kernel 独立内置参数实现

11.6.3.2.1 NOR

make menuconfig 配置内核，选如下配置（可以支持多选，但 ID 相同的 flash 只能选一个）

```
Device Drivers --->
<*> Memory Technology Device (MTD) support --->
Self-contained MTD device drivers --->
[*] Use SPI Nor Flash params built in kernel --->
--- Use SPI Nor Flash params built in kernel
[*] GD25Q127C 0xc84018
[ ] GD25Q256C 0xc84019    /*不能同时选择 I D相同的 flash*/
[*] GD25S512MD 0xc84019
```

相关配置路径如下：

```
kernel/drivers/mtd/devices/ingenic_sfc_v2/nor_device
├── Makefile
├── nor_device.c
└── nor_device.h

kernel/drivers/mtd/devices/Kconfig
```

如果需要添加自己型号的 flash，添加流程如下：

1. 根据 flash 型号， 添加 base params、cdt params、private params 参数。
   参考:kernel/drivers/mtd/devices/ingenic_sfc_v2/nor_device/nor_device.c*
2. 添加 config 配置
   参考:kernel/drivers/mtd/devices/Kconfig

11.6.3.2.2 NAND

默认使用内置 flash 参数

MTD 分区参数：

通过 device tree 中获取分区信息。

参考文件：kernel/arch/mips/boot/dts/ingenic/halley5_v20.dts

配置设备树：

```
&sfc {
        ingenic,use_ofpart_info = /bits/ 8 <1>; /*支持设备树 MTD 分区*/

        norflash@0 {
                partitions {
                        compatible = "fixed-partitions";
                        #address-cells = <1>;
                        #size-cells = <1>;

                        /* spi nor flash partition */
                        uboot@0 {
                                label = "uboot";
                                reg = <0x0000000 0x4 0000>;
                        };
                        kernel@40000 {
                                label = "kernel";
                                reg = <0x40000 0x300000>;
                        };
                        rootfs@360000 {
                                label = "rootfs";
                                reg = <0x360000 0xca0000>;
                        };
                };
```

```
        };
        ... ...
};
```

11.6.4 注意事项

1.Nand Flash ECC 方案

使用 nand flash 自带的硬件 ECC, 基于 MTD 框架，在内存中建立 BBT。

2.已经支持过的参数位置

NAND:

kernel/drivers/mtd/devices/ingenic_sfc_v2/nand_device/

NOR:

见烧录工具.

3.支持多 Die 的 SPI Nor Flash

spl:

暂不支持多 Die 切换

参数:

当 flash 参数使用烧录方式时，必须存放于 Die0

启动:

支持重启异常时 reset flash 切换 Die0

4.使用 SFC 的 PD 组 GPIO 时，要确保 PE 组 GPIO 配置成其它 function

注：

1. 当 PD 组和 PE 组同时配置为 sfc function 时，优先级 PE>PD

2. boot select 选择 usb 启动或者 sfc 1.8v 启动

3. 选择 sfc pD GPIO，修改设备树如下：

例：kernel/arch/mips/boot/dts/ingenic/halley5_v20.dts*

```
&sfc {
        . . .
        pinctrl-0 = <&sfc_pd_4bit>;
        . . .
}
```

12CPM 时钟电源复位接口

12.1 模块功能介绍

CPM 模块用于管理时钟和电源。包括：时钟控制，pll 控制，电源控制，复位控制。

CPM 驱动主要负责系统和模块的时钟电源管理，包括倍频，分频，时钟开关， 时钟源选择。

12.2 驱动位置

驱动源码所在位置：

```
drivers/clk/ingenic-v2$ tree
├── clk-bus.c
├── clk-bus.h
├── clk.c
├── clk-div.c
├── clk-div.h
├── clk.h
├── clk-m300.c
├── clk-pll.c
├── clk-pll.h
├── clk-x2000-v12.c
├── Kconfig
├── Makefile
├── power-gate.c
├── power-gate.h
```

12.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

设备树描述：

```
clock: clock-controller@0x10000000 {
        compatible = "ingenic,x2000-v12-clocks";
        reg = <0x10000000 0x100>;
        clocks = <&extclk>, <&rtcclk>;
        clock-names = "ext", "rtc_ext";
        #clock-cells = <1>;
        little-endian;
};

extclk: extclk {
        compatible = "ingenic,fixed-clock";
```

```
        clock-output-names ="ext";
        #clock-cells = <0>;
        clock-frequency  = <24000000>;
};

rtcclk: rtcclk {
        compatible = "ingenic,fixed-clock";
        clock-output-names ="rtc_ext";
        #clock-cells = <0>;
        clock-frequency = <32768>;
};
```

12.3.1 设备树默认配置

设备树默认编译会产生 clock，extclk，rtcclk 设备，配置 extclk 频率为 24Mhz：

```
extclk: extclk {
        clock-frequency  = <24000000>;
};
```

12.3.2 设备树自定义配置

12.4 内核编译配置

12.4.1 内核默认编译配置

使用 cpm 驱动内核不需要添加特殊配置，只依赖 soc 的配置选项 CONFIG_SOC_X2000_V12，相关代码在：drivers/clk/ingenic-v2/

12.4.2 内核自定义编译配置

12.5 设备节点生成

1. 生成的设备节点在

```
/sys/devices/platform/apb/10000000.clock-controller
```

2. 挂载 debugfs 到/mnt，/mnt/clk 下面的节点可以查看各个时钟的频率和状态
   a) cat clk_summary 可以查看所以时钟的父子关系，使能状态，频率等:

```
# cat clk_summary
   clock                         enable_cnt  prepare_cnt        rate   accuracy   phase
----------------------------------------------------------------------------------------
 rtc_ext                                  1            1       32768          0 0
    gate_rtc                              2            2       32768          0 0
 ext                                      2            2    24000000          0 0
    gate_ost                              2            2    24000000          0 0
    mux_dmic                              0            0    24000000          0 0
       gate_dmic                          0            0    24000000          0 0
          pd_mem_dmic                     0            0    24000000          0 0
    epll                                  0            0   300000000          0 0
       mux_i2s3                           0            0   300000000          0 0
          div_i2s3                        0            0   150000000          0 0
             ce_i2s3                      0            0   150000000          0 0
             gate_i2s3                    0            0   150000000          0 0
       mux_i2s2                           0            0   300000000          0 0
          div_i2s2                        0            0   150000000          0 0
             ce_i2s2                      0            0   150000000          0 0
             gate_i2s2                    0            0   150000000          0 0
       mux_i2s1                           0            0   300000000          0 0
          div_i2s1                        0            0   150000000          0 0
             ce_i2s1                      0            0   150000000          0 0
             gate_i2s1                    0            0   150000000          0 0
       mux_i2s0                           0            0   300000000          0 0
          div_i2s0                        0            0   150000000          0 0
             ce_i2s0                      0            0   150000000          0 0
             gate_i2s0                    0            0   150000000          0 0
             mux_spdif                    0            0   150000000          0 0
                gate_spdif                0            0   150000000          0 0
             mux_pcm                      0            0   150000000          0 0
                gate_pcm                  0            0   150000000          0 0
    mpll                                  9            9  1500000000          0 0
       mux_msc2                           1            1  1500000000          0 0
          div_msc2                        2            2    46875000          0 0
             gate_msc2                    1            1    46875000          0 0
                pd_mem_msc2               0            0    46875000          0 0
       mux_msc1                           1            1  1500000000          0 0
          div_msc1                        2            2    93750000          0 0
             gate_msc1                    1            1    93750000          0 0
                pd_mem_msc1               0            0    93750000          0 0
       mux_rsa                            0            0  1500000000          0 0
          div_rsa                         0            0   500000000          0 0
             gate_rsa                     0            0   500000000          0 0
       mux_isp                            1            1  1500000000          0 0
          div_isp                         2            2   300000000          0 0
       mux_pwm                            1            1  1500000000          0 0
          div_pwm                         2            2    93750000          0 0
             gate_pwm                     1            1    93750000          0 0
       mux_cim                            1            1  1500000000          0 0
          div_cim                         2            2    23809524          0 0
             gate_cim                     0            0    23809524          0 0
                pd_mem_cim                0            0    23809524          0 0
       mux_ssi                            0            0  1500000000          0 0
          div_ssi                         0            0     5882353          0 0
             gate_ssi0                    0            0     5882353          0 0
                pd_mem_ssi0               0            0     5882353          0 0
             gate_ssi1                    0            0     5882353          0 0
                pd_mem_ssi1               0            0     5882353          0 0
       mux_sfc                            1            1  1500000000          0 0
```

b) /mnt/clk 下,每个文件夹代表一个时钟，其中 mux_xxx 控制时钟源选择，div_xxx 控制时钟分频， gate_xxx 控制时钟的开关。pd_mem_xxx 控制每个模块 memory 的电源开关状态。power_xxx 控制模块的电源开关状态。

```
# ls
apll                gate_i2c2           mux_macphy
ce_i2s0             gate_i2c3           mux_macptp
ce_i2s1             gate_i2c4           mux_mactxphy
ce_i2s2             gate_i2c5           mux_mactxphy1
ce_i2s3             gate_i2s0           mux_msc0
clk_dump            gate_i2s1           mux_msc1
clk_orphan_dump     gate_i2s2           mux_msc2
clk_orphan_summary  gate_i2s3           mux_pcm
clk_summary         gate_intc           mux_pwm
div_ahb0            gate_isp0           mux_rsa
div_ahb2            gate_isp1           mux_sfc
div_apb             gate_lcd            mux_spdif
div_cim             gate_msc0           mux_ssi
div_cpu             gate_msc1           pd_mem_aes
div_cpu_l2c_x1      gate_msc2           pd_mem_audio
div_cpu_l2c_x2      gate_nemc           pd_mem_cim
div_ddr             gate_ost            pd_mem_ddr_ch0
div_i2s0            gate_otg            pd_mem_ddr_ch1
div_i2s1            gate_pcm            pd_mem_ddr_ch3
div_i2s2            gate_pdma           pd_mem_ddr_ch5
div_i2s3            gate_pwm            pd_mem_ddr_ch6
div_isp             gate_rot            pd_mem_ddr_top
div_l2c             gate_rsa            pd_mem_dmic
div_lcd             gate_rtc            pd_mem_dpu
div_macphy          gate_sadc           pd_mem_dsi
div_macptp          gate_scc            pd_mem_gmac0
div_mactxphy0       gate_sfc            pd_mem_gmac1
div_mactxphy1       gate_spdif          pd_mem_msc0
div_msc0            gate_ssi0           pd_mem_msc1
div_msc1            gate_ssi1           pd_mem_msc2
div_msc2            gate_tcu            pd_mem_nemc
div_pwm             gate_uart0          pd_mem_pdma
div_rsa             gate_uart1          pd_mem_pdma_sec
div_sfc             gate_uart2          pd_mem_rot
div_ssi             gate_uart3          pd_mem_sfc
epll                gate_uart4          pd_mem_ssi0
ext                 gate_uart5          pd_mem_ssi1
gate_aes            gate_uart6          pd_mem_uart0
gate_ahb0           gate_uart7          pd_mem_uart1
gate_apb0           gate_uart8          pd_mem_uart2
gate_arb            gate_uart9          pd_mem_uart3
gate_audio          gate_usbphy         pd_mem_uart4
gate_cim            mpll                pd_mem_uart5
gate_csi            mux_ahb0            pd_mem_uart6
gate_ddr            mux_ahb2            pd_mem_uart7
gate_dmic           mux_audio_ram       pd_mem_uart8
gate_dsi            mux_cim             pd_mem_uart9
gate_dtrng          mux_cpu_l2c         pd_mem_usb
gate_efuse          mux_ddr             power_felix
gate_felix          mux_dmic            power_helix
gate_gmac0          mux_i2s0            power_isp0
gate_gmac1          mux_i2s1            power_isp1
gate_hash           mux_i2s2            rtc_ext
gate_helix          mux_i2s3            sclka
gate_i2c0           mux_isp
gate_i2c1           mux_lcd
```

c） 进入到某一个时钟的文件夹内，cat clk_rate 可以查看时钟频率

```
# cd div_msc0/
#
# ls
clk_accuracy        clk_notifier_count  clk_rate
clk_enable_count    clk_phase
clk_flags           clk_prepare_count
#
#
# cat clk_rate
375000000
```

12.6 应用程序使用说明

13TCU 定时器单元

13.1 模块功能介绍

定时计数单元(TCU)芯片设计包含 8 个 tcu 通道分别是 0~7 标号，每个通道可以有三个时钟输入源 pwm(gpio0，gpio1)，extal，可以使用其中的任意组合进行功能的使用。

按照 spec 描述可以分为 7 种功能模式分别是：

常规模式：计数器在时钟的上升沿或是下降沿进行计数，也可以上升下降同时采集计数。

门模式：门为 0 时，计数器计数。

正交模式：计数器由于正交输入而计数。

方向模式：由输入信号决定计数器的递增还是递减。

pos 模式：由于上升沿或是下降沿，计数器开始从 0 开始计数。

捕获模式：计数器对周期进行计数并输出高电平时间和周期时间。

filter function (example for ch0)

TCU 通道	GPIO0	GPIO1
CH0	pwm0	pwm1
CH1	pwm2	pwm3
CH2	pwm4	pwm5
CH3	pwm6	pwm7
CH4	pwm8	pwm9
CH5	pwm10	pwm11
CH6	pwm12	pwm13
CH7	pwm14	pwm15

13.2 驱动位置

驱动源码所在位置

```
drivers/mfd/ingenic-tcu.c
```

13.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

TCU 控制器述：

```
tcu: tcu@0x10002000 {
        compatible = "ingenic,tcu";
        reg = <0x10002000 0x1000>;
        interrupt-parent = <&core_intc>;
        interrupt-names = "default";
        interrupts = <IRQ_TCU0>;
        status = "ok";
```

```
};
```

13.3.1 设备树默认配置

设备树默认编译会产生 tcu 设备

```
&pwm {
        status = "okay";
        pinctrl-names = "default","pwm15_pc";
        pinctrl-0 = <&pwm0_pd>;
        /*pinctrl-0 = <&pwm1_pc>; */
};
```

13.3.2 设备树自定义配置

用户可根据实际需求关闭 tcu 设备，将该节点配置为 disable。

```
&tcu {
          status = "disable";
};
```

13.4 内核编译配置

内核配置 MFD_INGENIC_TCU，配置说明如下：

```
Symbol: MFD_INGENIC_TCU [=y]
Type : boolean
Prompt: Ingenic tcu driver
        Location:
                -> Device Drivers
                        -> Multifunction device drivers
        Defined at drivers/mfd/Kconfig:417
        Depends on: HAS_IOMEM [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y])
        Selects: MFD_CORE [=y] && GENERIC_IRQ_CHIP [=y]
```

13.4.1 内核默认编译配置

内核默认配置 TCU 驱动，配置界面如下：

```
< > HTC PASIC3 LED/DS1WM chip support
[ ] HTC I2C PLD chip support
[ ] Support for Intel Atom SoC PMIC
< > Support for the Ingenic X2000_V12 and M300 SADC cor
< > Support for the Ingenic X2000_V12 and M300 SADC AUX
[*] Ingenic tcu driver
< > Kontron module PLD device
< > Marvell 88PM800
< > Marvell 88PM805
[ ] Marvell 88PM8606/88PM8607
[ ] Maxim Semiconductor MAX14577/77836 MUIC + Charger S
```

13.4.2 内核自定义编译配置

用户可根据实际需求去掉该驱动的配置。

13.5 设备节点生成

驱动加载成功后生成以下节点：

```
cd sys/devices/platform/apb/10002000.tcu/
disable enable power
driver modalias subsystem
driver_override of_node uevent
```

13.6 应用程序使用说明

1、使用主要用到的是 disable 和 enable

2、驱动中有一个配置的函数默认为 drivers/mfd/ingenic-tcu.c

```
/*This is a simple configuration test demo*/
static void ingenic_tcu_config_attr(int id,enum tcu_mode_sel mode_sel)
{
        int i = 0;
        g_tcu_chn[id].mode_sel = mode_sel;
        g_tcu_chn[id].irq_type = FULL_IRQ_MODE;
        ……
        for(i = 0; i < 20 ;i++)
                sws_pr_debug(&g_tcu_chn[id]);
}
```

说明：具体应用场景还没有，系统接口没有留太多，此函数的配置结合 spec 能够进行快速的验证功能,后续会留更多的系统配置接口。

在使用 DIRECTION_MODE、POS_MODE、CAPTURE_MODE、FILTER_MODE 模式之前请先配置对应 tcu 通道的 pwm0(gpio0)。

在使用 QUADRATURE_MODE 模式的时候请先配置对应 tcu 通道的 pwm0(gpio0)和 pwm1(gpio1)为正交信号。

3、使用过程就是把对应的通道写入 enable 该通道就开始工作

```
# cat enable

mode_sel :
0:GENERAL_MODE 1:GATE_MODE 2:DIRECTION_MODE 3:QUADRATURE_MODE
4:POS_MODE 5:CAPTURE_MODE 6:FILTER_MODE

####################example####################
## echo channel_id mode_sel > enable
## echo channel_id > disable

channel: 00 disable
channel: 01 disable
channel: 02 disable
```

```
channel: 03 disable
channel: 04 disable
channel: 05 disable
channel: 06 disable
channel: 07 disable

# echo 0 0 > enable
[ 164.485676] mode_id = 0
...
...
[                         165.896941]                                 ----------------------------------------count
NO.20------------------------------------------------
[ 165.896941]
[ 165.912304] -stop-----addr-b000201c-value-00000000----------
[ 165.918236] -mask-----addr-b0002030-value-00ff80fe----------
[ 165.924181] -enable---addr-b0002010-value-00000001----------
[ 165.930120] -flag-----addr-b0002020-value-00010000----------
[ 165.936048] -Control--addr-b000204c-value-00010004----------
[ 165.941990] -full-----addr-b0002040-value-0000ffff----------
[ 165.947925] -half- -addr-b0002044-value-00005000----------
[ 165.953874] -TCNT-----addr-b0002048-value-0000b5bf----------
[ 165.959811] -CAP reg_base-------value-00000000----------
[ 165.965399] -CAP_VAL register---value-00000000----------
```

4、把对应的通道写入 disable 就停止计数工作了

```
# echo 0 > disable
# cat disable
channel: 00 disable
channel: 01 disable
channel: 02 disable
channel: 03 disable
channel: 04 disable
channel: 05 disable
channel: 06 disable
channel: 07 disable
```

说明：tcu 使用的功能模式比较多，以至于没有留有太多的接口，但是内部逻辑都已经写好了，只需要按照 spec 中的不同模式的使用方式对 ingenic_tcu_config_attr 函数赋值就好了，每一类的宏也定义好了，能够方便的配置使用,不同的使用场景请修改相应的配置。

140ST 操作系统时钟

14.1 模块功能介绍

OST 是操作系统时钟，分为 global ost 和 core ost。global ost 是一个 64bit 的计数时钟，用于操作系统计时功能。core ost 是每个核有一个的 32bit 时钟，用于生成操作系统的时间片。

14.2 驱动位置

驱动源码所在位置：

```
/drivers/clocksource/ingenic_core_ost.c
```

14.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

设备树描述：

```
core_ost: core-ost@0x12000000 {
        compatible = "ingenic,core-ost";
        reg = <0x12000000 0x10000>, /*Global ost*/
                <0x12100000 0x10000>; /*Core ost*/
        interrupt-parent = <&cpuintc>;
        interrupt-names = "sys_ost";
        interrupts = <CORE_SYS_OST_IRQ>;
        cpu-ost-map =   <0 0x000>,
                        <1 0x100>;
};
```

14.3.1 设备树默认配置

默认编译会产生 ost 设备。

14.3.2 设备树自定义配置

14.4 内核编译配置

内核配置 CLKSRC_INGENIC_CORE_OST，配置说明如下：

```
Symbol: CLKSRC_INGENIC_CORE_OST [=y]
Type  : boolean
  Defined at drivers/clocksource/Kconfig:334
  Depends on: !ARCH_USES_GETTIMEOFFSET [=n]
  Selects: CLKSRC_OF [=y]
```

Selected by: SOC_X2000 [=n] && <choice> || SOC_X2000_V12 [=y] && <choice> || SOC_M300 [=n] && <choice>

14.4.1 内核默认编译配置

内核默认配置 ost 驱动

14.4.2 内核自定义编译配置

14.5 设备节点生成

/sys/devices/system/ clocksource

global ost 节点

/sys/devices/system/clockevents

core ost 节点

14.6 应用程序使用说明

15 INTC 中断控制器

15.1 模块功能介绍

intc 中断控制器控制处理器的中断源。64bit 中断源可以独立控制各个中断的打开和屏蔽。

15.2 驱动位置

驱动源码所在位置：

```
drivers/watchdog/ingenic_wtd.c
```

15.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

INTC 中断控制器描述：

```
cpuintc: interrupt-controller {
        ......
};

core_intc: core-intc@0x12300000 {
       ......
};
```

15.3.1 设备树默认配置

默认编译会产生 intc 设备

15.3.2 设备树自定义配置

15.4 内核编译配置

内核配置 IRQ_INGENIC_CPU，配置说明如下：

```
Symbol: IRQ_INGENIC_CPU [=y]
Type  : boolean
  Defined at drivers/irqchip/Kconfig:201
  Selects: IRQ_DOMAIN [=y]
  Selected by: SOC_X2000 [=n] && <choice> || SOC_X2000_V12 [=y] && <choice> || SOC_M300 [=n] && <choice>

Symbol: INGENIC_INTC_CHIP [=y]
Type  : boolean
  Defined at drivers/irqchip/Kconfig:207
  Depends on: MACH_XBURST2 [=y]
  Selects: IRQ_DOMAIN [=y]
```

```
Selected by: SOC_X2000 [=n] && <choice> || SOC_X2000_V12 [=y] && <choice> || SOC_M300 [=n] && <choice>
```

15.4.1 内核默认编译配置

内核默认配置 intc 驱动。

15.4.2 内核自定义编译配置

15.5 设备节点生成

驱动注册成功后生成设备节点

```
/sys/devices/platform/apb/10000000.clock-controller/subsystem/devices/12300000.core-intc
```

15.6 应用程序使用说明

查看中断：

```
cat /proc/interrupts
           CPU0       CPU1
  2:     804742     730919   XBurst2   2  xburst2-intc
  3:      51396      90444   XBurst2   3  jz-mailbox
  4:     158406     343976   XBurst2   4  core_timerevent
  8:          0          0    XBurst2-irqchip   0  as-dma
  9:      34037      24092   XBurst2-irqchip   1  13500000.otg, 13500000.otg, dwc2_hsotg:usb1
 11:          0          0   XBurst2-irqchip   3  pdma
 12:          0          0   XBurst2-irqchip   4  pdmad
 14:          0          0   XBurst2-irqchip   6  pwm-interrupt
 15:      32684      24604   XBurst2-irqchip   7  13440000.sfc
 26:          0          0    XBurst2-irqchip  18  13810000.vic
 27:         25        136    XBurst2-irqchip  19  13710000.vic
 28:          0          0    XBurst2-irqchip  20  13800000.isp
......
ERR:          0
```

16BT 蓝牙

16.1 模块简介

16.1.1 AP6256 芯片

x2000 使用的是 AP6256 芯片，AP6256 是基于 BCM4345C5 方案的集成 wifi 和 bluetooth 的功能模块。

- bluetooth 模块支持 HCI UART 接口、音频数据的 PCM 接口。
- bluetooth 符合蓝牙标准规范 5.0

16.1.2 1.2.GPIO 功能描述

bluetooth 模块

Name	I/O	Description
BT_REG_ON	I	Powerup/downinternalregulatorsusedbyBTsection
BT_WAKE	I	HOSTwake-upBluetoothdevice
BT_HOST_WAKE	0	Bluetoothdevicetowake-upHOST
UART_RTS_N	0	BluetoothUARTinterface
UART_TXD	0	BluetoothUARTinterface
UART_RXD	I	BluetoothUARTinterface
UART_CTS_N	I	BluetoothUARTinterface
PCM_OUT	0	PCMDataoutput
PCM_CLK	I/0	PCMclock
PCM_IN	I	PCMdatainput
PCM_SYNC	I/0	PCMsyncsignal

16.2 驱动源码位置

驱动源码所在位置：

```
kernel/drivers/misc/
├──bt_power_bluesleep.c
```

16.3 设备树配置

16.3.1 设备树默认配置

在板级设备树 halley5_v20.dts 中，进行如下默认配置：

1. 为蓝牙配置 uart

```
&uart0 {
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&uart0_pd>;
};
```

2. 配置电源管理相关 gpio

```
 bt_power {
        compatible = "ingenic,bt_power";
        ingenic,reg-on-gpio = <&gpd 20 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
        ingenic,wake-gpio = <&gpd 21 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
};
```

3. .配置音频相关的 pcm 接口

```
&as_be_baic {
       pinctrl-names = "default";
       pinctrl-0 =  <&baic4_pd>;
};
```

16.3.2 设备树自定义配置

用户可根据实际需求关闭相关配置。

16.4 内核编译配置

内核配置 BCM_4345C5_RFKILL，配置说明如下：

```
Symbol : BCM_4345C5_RFKILL[=y]
Type : tristate
Prompt : Bluetooth power control driver for BCM-4345C5 module
  Location :
    ->Device Drivers
      ->Misc devices
Defined a tdrivers/misc/Kconfig:499
Depends on : RFKILL[=y]
```

16.4.1 内核默认编译配置

内核默认配置打开该模块

16.4.2 内核自定义编译配置

用户可根据实际需求关闭该模块

16.5 设备节点生成

驱动加载成功后生成以下节点：

1. UART 节点

```
# ls dev/ttyS*
dev/ttyS0  dev/ttyS1  dev/ttyS3
```

2. 电源管理相关节点

```
# cat sys/class/rfkill/rfkill0/name
```

```
bluetooth
```

3. 音频相关节点

```
# ls dev/snd/
controlC0  pcmC0D1p   pcmC0D3p   pcmC0D5c   pcmC0D7c   pcmC0D9c
pcmC0D0p   pcmC0D2p   pcmC0D4p   pcmC0D6c   pcmC0D8c   timer
```

16.6 应用程序使用说明

16.6.1 模块供电

注意：首次系统上电必须 echo 1 sys/class/rfkill/rfkill0/state, 否则可能上电不成功**

```
# echo 1 > sys/class/rfkill/rfkill0/state
[ 1402.277128] restore_pin_status is not defined
```

16.6.2 通过串口获取蓝牙模块 ID

- 测试程序：OceanWing_bt_test_x2000
- 确保 bluetooth 供电
- 确保 UART 接口和蓝牙模块通信正常

```
# ./testsuite/OceanWing_bt_test_x2000 -d /dev/ttyS0



=======Read Successfully! Chip Version : BCM4345C5
```

16.6.3 蓝牙 uart 通路测试

搭建 bsa_server

测试程序：bsa_server

步骤：

1. 进入 adb shell

```
ser@user-HP-Compaq-8200:~$ adb shell
```

2. 执行 bsa_server,启动服务

```
# bsa_server -r 14 -p /firmware/BCM4345C5_003.006.006.0058.0135.hcd -u /var/run/ -d /dev/ttyS0
```

- -r 指定 baudrate 为` 3 M`*（UART 最大支持`3M`）
- -p 指定蓝牙固件路径
- -u 指定生成 bt 节点位置
- -d 设备

注：不能在串口直接执行 bsa_server,由于串口太慢，会导致 bsa_server 无法正常执行；测试需要通过 adb shell 执行 bsa_server。

3.bsa_server 服务成功后，会创建守护进程，产生两个 bt 节点

```
# ls var/run/bt*
var/run/bt-avk-fifo        var/run/bt-daemon-socket
```

16.6.4 蓝牙 pcm 通路测试

需要搭建蓝牙语音通话环境

16.6.5 基于 bsa server 开发参考内容

1. bsa_server 开发指南位置

```
packages/example/App/bluetooth_demo/BSA_GATT_Guide-v03.pdf
packages/example/App/bluetooth_demo/BSA_Simple_Guideline-v01.pdf
```

2. demo 源码目录

```
packages/example/App/bluetooth_demo/3rdparty/
```

3. 快速编译 app（详细参考 bsa_server 开发指南）

a) 配置交叉编译工具

```
# export PATH=prebuilts/toolchains/mips-gcc720-glibc226/bin:$PATH
```

b) 进入需要编译的 APP 的 build 下，以 app_manager 为例 */

```
# cd packages/example/App/bluetooth_demo
# cd 3rdparty/embedded/bsa_examples/linux/app_manager/build
```

c) 配置环境

```
# export MIPSGCC=mips-linux-gnu-gcc
```

d) 编译

```
# make CPU=mips clean
# make CPU=mips
```

e) build 目录下生成可执行程序

```
# ls mips/
app_manager  obj/
```

16.7 FAQ

16.7.1 bsa_server 不支持-lpm 参数

低功耗模式不支持

16.7.2 常见正常错误打印,不会影响正常使用

3D 命令不支持

```
BSA_trace 7827@ 01/01 00h:27m:42s:803ms: Num HCI Cmd Packets : 1 (0x01)
BSA_trace 7828@ 01/01 00h:27m:42s:803ms: Cmd Code : 0xfcb7 (VSC [Set TV2TV 3D])
BSA_trace 7829@ 01/01 00h:27m:42s:803ms: Status : Unknown HCI Command (0x01)
BSA_trace 7830@ 01/01 00h:27m:42s:803ms:   Raw HCI Data Received:
BSA_trace 7831@ 01/01 00h:27m:42s:803ms:     0000: 0e 04 01 b7 fc 01
BSA_trace 7832@ 01/01 00h:27m:42s:803ms: >>
BSA_trace 7833@ 01/01 00h:27m:42s:803ms: [3DTV]bsa_sv_dm_send_3dtv_vsc_cmpl_cback
BSA_trace 7834@ 01/01 00h:27m:42s:803ms: [3DTV]ERROR bsa_sv_dm_send_3dtv_vsc_cmpl_cback FAIL
BSA_trace 7835@ 01/01 00h:27m:42s:803ms: [3DTV]bsa_sv_dm_send_3dtv_vsc_cmpl_cback RETURN:8
BSA_trace 7836@ 01/01 00h:27m:42s:803ms: bsa_sv_dm_send_set_config_complete return status:0x0
```

检查低功耗节点不存在

```
BSA_trace 7681@ 01/01 00h:27m:42s:795ms: HCILP_Disable
BSA_trace 7682@ 01/01 00h:27m:42s:795ms: BTM: BTM_VendorSpecificCommand: Opcode: 0xFC27, ParamLen: 12.
BSA_trace 7683@ 01/01 00h:27m:42s:795ms: UPIO_set_bluesleep_proto : 0
BSA_trace 7684@ 01/01 00h:27m:42s:795ms: Fail to open /proc/bluetooth/sleep/proto
```

17WDT 看门狗

17.1 模块功能介绍

看门狗定时器用于在处理器受噪声和系统错误等故障干扰时恢复处理器，看门狗定时器可以产生复位信号。看门狗使用的是 rtc 提供的时钟源，通过软件可以分频为 1，4，16，64，256 和 1024，并且拥有 16bit 的计数寄存器，除此之外还支持半中断处理。

17.2 驱动源码位置

驱动源码所在位置：

```
drivers/watchdog/ingenic_wtd.c
```

17.3 设备树配置

设备树所在位置

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

WDT 控制器描述

```
watchdog: watchdog@0x10002000 {
        compatible = "ingenic,watchdog";
        reg = <0x10002000 0x40>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_TCU0>;
        status = "ok";
}
```

17.3.1 设备树默认配置

默认编译会产生 WDT 设备。

17.3.2 设备树自定义配置

用户可根据 shiji 需求关闭 WDT 设备，将该节点配置为 disable：

```
&watchdog {
          status = "disable";
};
```

17.4 内核编译配置

内核配置 INGENIC_WDT，配置说明如下：

```
Symblo:  INGENIC_WDT[=y]
Type: tristate
Prompt: Ingenic ingenic SoC hardware watchdog
    Location:
        -> Device Drivers
```

```
        ->Watchdog Timer Support (WATCHDOG [=y])
  Defined at drivers/watchdog/Kconfig:1212
  Depends on: WATCHDOG [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y])
  Selects: WATCHDOG_CORE [=y]
```

17.4.1 内核默认编译配置

内核默认配置 WDT 看门狗，配置界面如下：

```
--- Watchdog Timer Support
-*-   WatchDog Timer Driver Core
[ ]   Disable watchdog shutdown on close
      *** Watchdog Device Drivers ***
< >   Software watchdog
< >   Watchdog device controlled through GPIO-line
< >   Xilinx Watchdog timer
< >   Cadence Watchdog Timer
< >   Synopsys DesignWare watchdog
< >   Max63xx watchdog
<*>   Ingenic ingenic SoC hardware watchdog
< >   BCM7038 Watchdog
< >   Imagination Technologies PDC Watchdog Timer
< >   MEN A21 VME CPU Carrier Board Watchdog Timer
      *** USB-based Watchdog Cards ***
< >   Berkshire Products USB-PC Watchdog
```

17.4.2 内核自定义编译配置

用户可根据实际需求，去掉该驱动的配置。

17.5 设备节点生成

驱动注册成功后生成设备节点

```
/dev/watchdog
```

17.6 应用程序使用说明

发布的 SDK 中 busybox 会默认配置 watchdog 测试应用，以下为测试应用的说明：

```
watchdog [-t N[ms]] [-T N[ms]] [-F] DEV

    Periodically write to watchdog device DEV

    Options:

            -T N Reboot after N seconds if not reset (default 60)
            -t N Reset every N seconds (default 30)
            -F Run in foreground
    Use 500ms to specify period in milliseconds
```

执行以下命令测试 watchdog：

1、设置喂狗的周期

```
watchdog -t 3 /dev/watchdog0  /*每隔 3s 执行一次喂狗操作 ping*/
```

2、设置超时重启

```
watchdog -t 5 -T 3 /dev/watchdog0 /*3s后复位（每隔5s执行一次喂狗，3s没喂狗则reset）*/
```

18PDMA 控制器

18.1 模块功能介绍

可编程 DMA 控制器（PDMAC）专门用于在片上外设（MSC，AIC，UART 等），外部存储器和存储器映射的外部设备之间智能传输数据，支持多达 32 个独立的 DMA 通道。

18.2 驱动位置

驱动源码所在位置：

```
drivers/dma/ingenic$ tree .
├── ingenic_dma.c
├── ingenic_dma.h
└── Makefile
```

18.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

PDMA 控制器描述：

```
pdma: dma@13420000 {
        compatible = "ingenic,x2000-pdma";
        reg = <0x13420000 0x10000>;
        interrupt-parent = <&core_intc>;
        interrupt-names = "pdma", "pdmad";
        interrupts = <IRQ_PDMA>, <IRQ_PDMAD>;
        #dma-channels = <32>;
        #dma-cells = <1>;
        ingenic,reserved-chs = <0x3>;
};
```

18.3.1 设备树默认配置

默认编译会产生 pdma 设备

```
&pdma {
        status = "okay";
};
```

18.3.2 设备树自定义配置

用户可根据实际需求关闭 PDMA 设备，可以将该节点配置为 disable。

```
&pdma {
        status = "disable";
};
```

18.4 内核编译配置

内核配置 INGENIC_PDMAC，配置说明如下：

```
Symbol: INGENIC_PDMAC [=y]
Type : boolean
Prompt: Ingenic programmable dma controller (Of Driver)
      Location:
          -> Device Drivers
              -> DMA Engine support (DMADEVICES [=y])
      Defined at drivers/dma/Kconfig:280
      Depends on: DMADEVICES [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y])
      Selects: DMA_VIRTUAL_CHANNELS [=y] && DMA_ENGINE [=y]
      Selected by: SND_ASOC_PDMA [=n] && SOUND [=n] && !M68K && !UML && SND [=n] && SND_SOC [=n] &&
SND_ASOC_INGENIC [=n] && SND_ASOC_INGENIC_AS_V1 [=n]
```

18.4.1 内核默认编译配置

内核默认配置 PDMA 驱动，

配置界面如下：

```
--- DMA Engine support
[ ]   DMA Engine debugging
      *** DMA Devices ***
< >   Freescale eDMA engine support
< >   IMG MDC support
< >   Intel integrated DMA 64-bit support
[*]   Ingenic programmable dma controller (Of Driver)
< >   Synopsys DesignWare AHB DMA platform driver
      *** DMA Clients ***
[ ]   Async_tx: Offload support for the async_tx api
< >   DMA Test client
```

18.4.2 内核自定义编译配置

1. 用户可根据实际需求，去掉该驱动的配置。
2. 驱动测试配置，配置说明如下：

```
Symbol: DMATEST [=y]
Type : tristate
Prompt: DMA Test client
      Location:
          -> Device Drivers
              -> DMA Engine support (DMADEVICES [=y])
      Defined at drivers/dma/Kconfig:561
      Depends on: DMADEVICES [=y] && DMA_ENGINE [=y]
```

配置界面如下：

```
--- DMA Engine support
[ ]   DMA Engine debugging
      *** DMA Devices ***
< >   Freescale eDMA engine support
< >   IMG MDC support
< >   Intel integrated DMA 64-bit support
[*]   Ingenic programmable dma controller (Of Driver)
< >   Synopsys DesignWare AHB DMA platform driver
      *** DMA Clients ***
[ ]   Async_tx: Offload support for the async_tx api
<*>   DMA Test client
```

18.5 设备节点生成

18.5.1 控制器节点

驱动加载成功后生成以下控制器节点：

```
# cd /sys/devices/platform/ahb2/13420000.dma/dma/
# ls
dma0chan0 dma0chan14 dma0chan2 dma0chan25 dma0chan30 dma0chan8
dma0chan1 dma0chan15 dma0chan20 dma0chan26 dma0chan31 dma0chan9
dma0chan10 dma0chan16 dma0chan21 dma0chan27 dma0chan4
dma0chan11 dma0chan17 dma0chan22 dma0chan28 dma0chan5
dma0chan12 dma0chan18 dma0chan23 dma0chan29 dma0chan6
dma0chan13 dma0chan19 dma0chan24 dma0chan3 dma0chan7
```

18.5.2 测试程序节点

若配置驱动测试，生成以下测试节点：

```
# cd /sys/module/dmatest/parameters/
# ls
channel noverify threads_per_chan xor_sources
device pq_sources timeout
iterations run verbose
max_channels test_buf_size wait
```

18.6 应用程序使用说明

18.6.1 dmatest 测试程序

1. 进入到测试程序的节点

```
# echo dma0chan31 > /sys/module/dmatest/parameters/channel
# echo 2000 > /sys/module/dmatest/parameters/timeout
# echo 1 > /sys/module/dmatest/parameters/iterations
# echo 1 > /sys/module/dmatest/parameters/run
```

2. 测试结果

```
[ 148.309057] dmatest: Started 1 threads using dma0chan31
[ 148.309825] dmatest: dma0chan31-copy: summary 1 tests, 0 failures 1355 iops 10840 KB/s (0)
```

18.6.2 SSI 驱动程序测试

PDMA 驱动遵循内核标准的 dmaengine 驱动框架，可作为 provider 为其他驱动程序提供 dma 传输通道，这里以 SSI 驱动程序为例，介绍如何配置。

18.6.2.1 设备树配置

```
spi0: spi0@0x10043000 {
    compatible = "ingenic,spi";
    reg = <0x10043000 0x1000>;
    interrupt-parent = <&core_intc>;
    interrupts = <IRQ_SSI0>;
    dmas = <&pdma INGENIC_DMA_TYPE(INGENIC_DMA_REQ_SSI0_TX)>,
           <&pdma INGENIC_DMA_TYPE(INGENIC_DMA_REQ_SSI0_RX)>;
    dma-names = "tx", "rx";
    #address-cells = <1>;
    #size-cells = <0>;
    status = "disabled";
};
```

其中，dmas 和 dma-names 节点为 dma 相关配置

```
dmas = <&pdma INGENIC_DMA_TYPE(INGENIC_DMA_REQ_SSI0_TX)>,
       <&pdma INGENIC_DMA_TYPE(INGENIC_DMA_REQ_SSI0_RX)>;
dma-names = "tx", "rx";
```

dmas 表示使用 PDMA 提供的对应传输通道

dma-names 为一个标志，用于在 SSI 驱动程序中申请 dma 传输通道

18.6.2.2 驱动程序使用流程

由于使用的是内核标准的 dmaengine 驱动框架，所有 consumer 端驱动程序均使用相同的 api，使用流程基本相同。

在 SSI 驱动程序中使用流程如下：

1. 申请 dma channel

```
ingspi->txchan = dma_request_slave_channel_reason(dev, "tx");
ingspi->rxchan = dma_request_slave_channel(dev, "rx");
```

2. 配置 dma channel 的参数

```
dmaengine_slave_config(txchan, &tx_config);
dmaengine_slave_config(rxchan, &rx_config);
```

3. 获取传输描述符

```
txdesc = dmaengine_prep_slave_sg(
                    txchan,
                    t->tx_sg.sgl, t->tx_sg.nents,
                    DMA_DEV_TO_MEM, DMA_PREP_INTERRUPT | DMA_CTRL_ACK);
```

```
rxdesc = dmaengine_prep_slave_sg(
                    rxchan,
                    t->rx_sg.sgl, t->rx_sg.nents,
                    DMA_DEV_TO_MEM, DMA_PREP_INTERRUPT | DMA_CTRL_ACK);
```

4. 提交并启动传输

```
dmaengine_submit(txdesc);
dmaengine_submit(rxdesc);
dma_async_issue_pending(rxchan);
dma_async_issue_pending(txchan);
```

19SADC 控制器

19.1 模块功能介绍

X2000 芯片的 A/D 转换模块是 10 位的高精度模数转换器，支持 6 路模拟输入。

19.2 内核源码路径

```
kernel-4.4.94/drivers/mfd/ingenic_adc_v13.c
kernel-4.4.94/drivers/mfd/ingenic_adc_aux.c
```

19.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

ADC 模块设备树：

```
sadc: sadc@10070000 {
        compatible = "ingenic,x2000-v12-sadc";
        reg = <0x10070000 0x32>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_SADC>;
        interrupt-controller;
        status = "ok";
    };
```

19.3.1 设备树默认配置

默认编译会产生 sadc 设备

19.3.2 设备树自定义配置

用户可根据需求在板级设备树中添加 sadc，并使能

```
&sadc {
    status = "okay";
};
```

19.4 内核编译配置

19.4.1 内核默认编译配置

配置内核并编译 sadc 驱动：

```
Symbol: MFD_INGENIC_SADC_V13 [=y]
Type  : tristate
Prompt: Support for the Ingenic SADC core
  Location:
```

```
    -> Device Drivers
      -> Multifunction device drivers
  Defined at drivers/mfd/Kconfig:417
  Depends on: HAS_IOMEM [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y])
  Selects: MFD_CORE [=y]

Symbol: MFD_INGENIC_SADC_AUX [=y]
Type  : tristate
Prompt: Support for the Ingenic SADC AUX
  Location:
    -> Device Drivers
      -> Multifunction device drivers
  Defined at drivers/mfd/Kconfig:425
  Depends on: HAS_IOMEM [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y])
  Selects: MFD_CORE [=y]
```

19.4.2 内核自定义编译配置

SADC 配置界面如下：

```
[ ] Support for Intel Atom SoC PMIC
<*> Support for the Ingenic SADC core
<*> Support for the Ingenic SADC AUX
[*]   Ingenic sadc aux reference voltage 1.8V.
[*]   ingenic sadc aux 10bit
[*] Ingenic tcu driver
[ ] Ingenic tcu driver v1
```

19.5 设备节点生成

设备加载成功后会在/dev 目录下生成如下节点：

```
# ls /dev/ingenic_adc_aux_*
/dev/ingenic_adc_aux_0  /dev/ingenic_adc_aux_2  /dev/ingenic_adc_aux_4
/dev/ingenic_adc_aux_1  /dev/ingenic_adc_aux_3  /dev/ingenic_adc_aux_5
#
```

19.6 应用程序使用说明

19.6.1 应用程序源码位置

```
packages/example/Sample/sadc
```

19.6.1.1 命令行及参数示意

应用程序默认存放在/testsuite/sadc_sample

```
cd  /testsuite/sadc_sample
./sadc_sample adc_num
```

其中的 adc_num 指/dev 下 adc 设备节点号；在 x2000 芯片中，adc_num 可取的值为：0,1,2,3,4,5.

20RTC 控制器

20.1 模块功能介绍

实时时钟（RTC）单元可以在芯片主电源打开或主电源关闭但 RTC 电源仍然打开的情况下运行。在这种情况下，RTC 电源域仅消耗几微瓦的功率。RTC 包含实时，定时告警逻辑以及断电和唤醒控制逻辑。

20.2 驱动位置

驱动源码所在位置

```
drivers/rtc/rtc-ingenic.c
drivers/rtc/rtc-ingenic.h
```

20.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

RTC 控制器描述：

```
rtc: rtc@0x10003000 {
        compatible = "ingenic,rtc";
        reg = <0x10003000 0x4c>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_RTC>;
        system-power-controller;
        power-on-press-ms = <1000>;
        status = "ok";
};
```

20.3.1 设备树默认配置

设备树默认编译会产生 RTC 设备。

20.3.2 设备树自定义配置

用户可根据实际需求关闭 RTC 设备，在板级.dts 中将该节点配置为 disabled。

```
&rtc {
        status = "disabled";
};
```

20.4 内核编译配置

内核配置 RTC_DRV_INGENIC，配置说明如下：

```
Symbol: RTC_DRV_INGENIC [=y]
Type : tristate
Prompt: INGENIC RTC
```

```
Location:
  -> Device Drivers
    -> Real Time Clock (RTC_CLASS [=y])
Defined at drivers/rtc/Kconfig:142
Depends on: RTC_CLASS [=y]
```

20.4.1 内核默认编译配置

内核默认配置 RTC 驱动，配置界面如下：

```
--- Real Time Clock
[*]   Set system time from RTC on startup and resume
(rtc0)  RTC used to set the system time
[*]   Set the RTC time based on NTP synchronization
(rtc0)  RTC used to synchronize NTP adjustment
[ ]   RTC debug support
      *** RTC interfaces ***
[*]   /sys/class/rtc/rtcN (sysfs)
[*]   /proc/driver/rtc (procfs for rtcN)
[*]   /dev/rtcN (character devices)
[ ]     RTC UIE emulation on dev interface
< >   Test driver/device
<*>   INGENIC RTC
[ ]     Ingenic RTC suspend test
      *** I2C RTC drivers ***
< >   Abracon AB-RTCMC-32.768kHz-B5ZE-S3
< >   Abracon ABx80x
< >   Dallas/Maxim DS1307/37/38/39/40, ST M41T00, EPSON RX-8025
< >   Dallas/Maxim DS1374
< >   Dallas/Maxim DS1672
< >   Dallas/Maxim DS3232
< >   Haoyu Microelectronics HYM8563
< >   Maxim MAX6900
< >   Ricoh R2025S/D, RS5C372A/B, RV5C386, RV5C387A
< >   Intersil ISL1208
< >   Intersil ISL12022
< >   Intersil ISL12057
< >   Xicor/Intersil X1205
< >   NXP PCF2127
< >   NXP PCF8523
< >   Philips PCF8563/Epson RTC8564
< >   nxp PCF85063
< >   Philips PCF8583
< >   ST M41T62/65/M41T80/81/82/83/84/85/87 and compatible
< >   TI BQ32000
< >   Seiko Instruments S-35390A
< >   Ramtron FM3130
< >   Epson RX-8581
< >   Epson RX-8025SA/NB
< >   EM Microelectronic EM3027
< >   Micro Crystal RTC
< >   Micro Crystal RV8803
      *** SPI RTC drivers ***
```

20.4.2 内核自定义编译配置

1. 用户可根据实际需求去掉该驱动的配置。

2. RTC 测试时配置 SUSPEND_TEST 和 SUSPEND_ALARM_TIME，配置说明如下：

```
Symbol: SUSPEND_TEST [=y]
```

```
Type : boolean
Prompt: auto suspend test
     Location:
          -> Machine selection
               -> SOC type (SOC_TYPE [=n])
Prompt: Ingenic RTC suspend test
     Location:
          -> Device Drivers
               -> Real Time Clock (RTC_CLASS [=y])
                    -> INGENIC RTC (RTC_DRV_INGENIC [=y])
     Defined at arch/mips/xburst/Kconfig:61
     Depends on: SOC_TYPE [=n]
```

```
Symbol: SUSPEND_ALARM_TIME [=60]
Type : integer
Prompt: suspend alarm time(second)
     Location:
          -> Machine selection
               -> SOC type (SOC_TYPE [=n])
                    -> auto suspend test (SUSPEND_TEST [=y])
Prompt: Ingenic RTC suspend alarm time Unit of second
     Location:
          -> Device Drivers
               -> Real Time Clock (RTC_CLASS [=y])
                    -> INGENIC RTC (RTC_DRV_INGENIC [=y])
                         -> Ingenic RTC suspend test (SUSPEND_TEST [=y])
     Defined at arch/mips/xburst/Kconfig:65
     Depends on: SOC_TYPE [=n] && SUSPEND_TEST [=y]
```

配置界面如下：

```
--- INGENIC RTC
[*]   Ingenic RTC suspend test
(60)    Ingenic RTC suspend alarm time Unit of second
```

20.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/rtc0
```

20.6 应用程序使用说明

20.6.1 定时唤醒测试

1. 首先在内核编译配置中进行 RTC 测试配置。

2. 重新编译内核并启动
3. 执行命令

```
echo mem > sys/power/state
```

休眠 60 秒后自动唤醒一（测试点为定时和计时准确以及唤醒功能）

21EFUSE 接口

21.1 模块功能介绍

EFUSE 模块提供了读写 efuse 各个数据段的接口。可以查看 efuse 中保存的 cpu 相关信息或者写入用户自定义的信息。

21.2 驱动位置

驱动源码所在位置

```
drivers/misc/jz_efuse_x2000.c
```

21.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

RTC 控制器描述：

```
efuse: efuse@0x13540000 {
                        compatible = "ingenic,x2000-efuse";
                        reg = <0x13540000 0x10000>;
                        status = "okay";
};
```

21.3.1 设备树默认配置

设备树默认编译会产生 EFUSE 设备。

21.3.2 设备树自定义配置

```
&efuse{
        status = "disabled";
        ingenic,efuse-en-gpio = <&gpd 19 GPIO_ACTIVE_LOW  INGENIC_GPIO_NOBIAS>;
};
```

21.4 内核编译配置

内核配置 JZ_EFUSE_X2000，配置说明如下：

```
Symbol: JZ_EFUSE_X2000 [=y]
Type  : boolean
Prompt: JZ EFUSE X2000 Driver
Location:
  -> Device Drivers
```

```
      -> Misc devices
 Defined at drivers/misc/Kconfig:566
```

当需要向 efuse 中写入数据时，需要打开宏 INGENIC_EFUSE_WRITABLE。该宏配置方法如下。

```
Symbol: INGENIC_EFUSE_WRITABLE [=y]
Type  : boolean
Prompt: Ingenic Efuse X2000 Writable
Location:
  -> Device Drivers
      -> Misc devices
 Defined at drivers/misc/Kconfig:575
Depends on: INGENIC_EFUSE_X2000 [=y]
```

21.4.1 内核默认编译配置

配置界面如下：

```
< > Analog Devices Digital Potentiometers
< > Dummy IRQ handler
< > Integrated Circuits ICS932S401
< > Enclosure Services
< > Medfield Avago APDS9802 ALS Sensor module
< > Intersil ISL29003 ambient light sensor
< > Intersil ISL29020 ambient light sensor
< > Taos TSL2550 ambient light sensor
< > ROHM BH1780GLI ambient light sensor
< > BH1770GLC / SFH7770 combined ALS - Proximity sensor
< > APDS990X combined als and proximity sensors
< > Honeywell HMC6352 compass
< > Dallas DS1682 Total Elapsed Time Recorder with Alarm
< > Texas Instruments DAC7512
< > BMP085 digital pressure sensor on I2C
< > BMP085 digital pressure sensor on SPI
< > FSA9480 USB Switch
<*> Bluetooth power control driver for BCM-4345C5 module
< > Lattice ECP3 FPGA bitstream configuration via SPI
[ ] Generic on-chip SRAM driver
[ ] JZ RSA Driver
[ ] Linux pmem allocator
[*] Ingenic Efuse X2000 Driver
[ ] Ingenic ak5811 Driver
[*] Ingenic Efuse X2000 Writable
[ ] sy7734 driver
[ ] ingenic rmem driver
```

21.4.2 内核自定义编译配置

用户可根据实际需求添加该模块。该模块默认不配置。

21.5 设备节点生成

驱动加载成功后会在/sys/devices/platform/ahb2/13540000.efuse/efuse_rw/目录下生成以下文件：

```
chipid
chipkey
custid0
custid1
custid2
hideblk
nku
prt
socinfo
trim0
trim1
trim2
userkey0
userkey1
```

生成的每个文件都对应 efuse 的一个段，具体每个段的作用，请参考 pm 手册。

21.6 应用程序使用说明

21.6.1 读取 efuse 各个段的数据

```
cd /sys/devices/platform/ahb2/13540000.efuse/efuse_rw/
cat chipid
cat ...
cat socinfo
```

其中的段 userkey,chidkey,nku 和加密相关，不能通过该方式读。

21.6.2 写数据到 efuse 的各个段

写 efuse 时，数据要以 16 进制字符串的方式写入，不加 0x 前缀。以 socinfo 为例，写数据 0x2080020800 到 efuse 中。

执行指令

```
cd /sys/devices/platform/ahb2/13540000.efuse/efuse_rw/
echo  2080020800 > socinfo
```

其中的段 userkey,chidkey,nku 和加密相关，不能通过该方式读。

21.7 注意事项

efuse 只能写入一次，重复写入可能会造成芯片产生不可预知的错误，写入时需谨慎。

22GPIO 通用 IO 接口

22.1 模块功能介绍

gpio 一般在进行开发板设计的时候就已经固定好了， 有的 gpio 只能作为设备复用功能管脚，有的 gpio 作为普通的输入输出和中断检测功能， 对于固定设备复用的功能管脚在以下文件中定义：

```
arch/mips/boot/dts/ingenic/x2000-v12-pinctrl.dtsi
```

在 arch/mips/boot/dts/ingenic/"板级".dts 会根据驱动配置， 选中相应的设备功能管脚。内核的 gpio 驱动基于 gpio 子系统实现，很方便的进行 gpio 控制。

22.2 驱动位置

驱动源码所在位置

```
drivers/gpio
```

22.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12-pinctrl.dtsi
```

部分 GPIO 配置如下：

```
&pinctrl {
        uart0_pin: uart0-pin {
                uart0_pd: uart0-pd {
                        ingenic,pinmux = <&gpd 23 26>;
                        ingenic,pinmux-funcsel = <PINCTL_FUNCTION2>;
                };
        };
        uart1_pin: uart1-pin {
           ……
        };

        uart2_pin: uart2-pin {
            ……
        };

        uart3_pin: uart3-pin {
            ……
}
```

22.3.1 设备树默认配置

halley5_v20.dts 中规定了各个 GPIO 的默认配置，并将 PA 组 GPIO 配置为 1.8V（PA 组 GPIO 可配置

为 1.8V 或 3.3V）

```
&pinctrl {
        ingenic,gpa_voltage = <GPIO_VOLTAGE_1V8>;
};
```

22.3.2 设备树自定义配置

22.3.2.1 配置 pincfg 属性

当 gpio 作为设备功能复用时，可以通过"ingenic,pincfg" 指定某个 pin 脚的附加属性,

- 上拉
- 下拉
- 高阻
- 驱动能力
- SCHMITT
- SLEW_RATE

具体语法：

```
ingenic,pincfg = <gpio_group start_pin end_pin pincfgs>
```

其中 pincfgs 可以是以下形式:

```
#define PINCTL_CFG_BIAS_DISABLE                 1
#define PINCTL_CFG_BIAS_HIGH_IMPEDANCE          2
#define PINCTL_CFG_BIAS_PULL_DOWN               3
#define PINCTL_CFG_BIAS_PULL_PIN_DEFAULT        4
#define PINCTL_CFG_BIAS_PULL_UP                 5
#define PINCTL_CFG_DRIVE_STRENGTH               9
#define PINCTL_CFG_FILTER                       10
#define PINCTL_CFG_INPUT_SCHMITT_ENABLE         11
#define PINCTL_CFG_SLEW_RATE                    17
```

```
* PINCFG_PACK(PINCTL_CFG_DRIVE_STRENGTH, 7)     # value: 0 ~ 7
* PINCFG_PACK(PINCTL_CFG_BIAS_DISABLE,1)
* PINCFG_PACK(PINCTL_CFG_PULL_DOWN, 1)          # 使能下拉
* PINCFG_PACK(PINCTL_CFG_PULL_UP, 1)            # 使能上拉
* PINCFG_PACK(PINCTL_CFG_INPUT_SCHMITT_ENABLE, 1)
* PINCFG_PACK(PINCTL_CFG_SLEW_RATE, 1)
```

例如：

```
mac0_rgmii_p1_normal: mac0-rgmii-p1-normal {
        ingenic,pinmux = <&gpc 1 5>, <&gpc 10 10>, <&gpc 12 15>;
        ingenic,pinmux-funcsel = <PINCTL_FUNCTION1>;
        ingenic,pincfg = <&gpc 1 5 PINCFG_PACK(PINCTL_CFG_DRIVE_STRENGTH, 7)>, \
                         <&gpc 10 10 PINCFG_PACK(PINCTL_CFG_DRIVE_STRENGTH, 7)>,\
```

```
                <&gpc 12 15 PINCFG_PACK(PINCTL_CFG_DRIVE_STRENGTH, 7)>;
};
```

22.3.2.2 配置 pincfg 属性

当定义一个普通 GPIO 时，也可以配置 GPIO 的属性。具体如下:

```
#define INGENIC_GPIO_BIAS_MASK          0x7
#define INGENIC_GPIO_BIAS_SFT           0
#define INGENIC_GPIO_NOBIAS             (0 << INGENIC_GPIO_BIAS_SFT)
#define INGENIC_GPIO_PULLEN             (1 << INGENIC_GPIO_BIAS_SFT)
#define INGENIC_GPIO_PULLUP             (2 << INGENIC_GPIO_BIAS_SFT)
#define INGENIC_GPIO_PULLDOWN           (3 << INGENIC_GPIO_BIAS_SFT)
#define INGENIC_GPIO_HIZ                (4 << INGENIC_GPIO_BIAS_SFT)

#define INGENIC_GPIO_DS_SFT             3
#define INGENIC_GPIO_DS_MSK             0x7
#define INGENIC_GPIO_DS_0               (0 << INGENIC_GPIO_DS_SFT)
#define INGENIC_GPIO_DS_1               (1 << INGENIC_GPIO_DS_SFT)
#define INGENIC_GPIO_DS_2               (2 << INGENIC_GPIO_DS_SFT)
#define INGENIC_GPIO_DS_3               (3 << INGENIC_GPIO_DS_SFT)
#define INGENIC_GPIO_DS_4               (4 << INGENIC_GPIO_DS_SFT)
#define INGENIC_GPIO_DS_5               (5 << INGENIC_GPIO_DS_SFT)
#define INGENIC_GPIO_DS_6               (6 << INGENIC_GPIO_DS_SFT)
#define INGENIC_GPIO_DS_7               (7 << INGENIC_GPIO_DS_SFT)

#define INGENIC_GPIO_SLEW_RATE_SFT      6
#define INGENIC_GPIO_SLEW_RATE          (1 << INGENIC_GPIO_SLEW_RATE_SFT)

#define INGENIC_GPIO_SCHMITT_SFT        7
#define INGENIC_GPIO_SCHMITT            (1 << INGENIC_GPIO_SCHMITT_SFT)
```

当需要应用多个属性到某一个 GPIO 时，使用值或即可，例如，将 GPIO 配置为上拉、SCHMITT、驱动能力 6

```
ingenic,lcd-pwm-gpio = <&gpc 1 GPIO_ACTIVE_LOW (INGENIC_GPIO_PULLUP | INGENIC_GPIO_SCHMITT | INGENIC_GPIO_DS_6 )>;
```

定义通用 GPIO 示例:

```
bt_power {
        compatible = "ingenic,bt_power";
        ingenic,reg-on-gpio = <&gpd 20 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
        ingenic,wake-gpio = <&gpd 21 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
};
```

22.3.2.3 系统休眠 GPIO 配置

在系统休眠时，针对每个开发板的 GPIO 配置不一样，想降低系统休眠功耗，需要对 GPIO 进行输入上下拉或者输出高地等配置。

在板极.dts 文件中，可以通过配置以下的 gpio 各组 gpio 属性，让系统在休眠时，处于相应的状态.例如：配置 PA 组(0 - 10)的 pin 为上拉状态，(11 - 15)为输出高

```
&gpa {
        ingenic,gpio-sleep-pullup       = <0 1 2 3 4 5 6 7 8 9 10>;
        ingenic,gpio-sleep-pulldown     = <>;
        ingenic,gpio-sleep-hiz          = <>;
        ingenic,gpio-sleep-low          = <>;
        ingenic,gpio-sleep-high         = <11 12 13 14 15>;
};
```

按照实际情况配置以下属性：

```
/* Board Sleep GPIO configuration. */
/* <0 1 2 3 .... 31>, set one of the pin to state.*/
&gpa {
        ingenic,gpio-sleep-pullup       = <>;
        ingenic,gpio-sleep-pulldown     = <>;
        ingenic,gpio-sleep-hiz          = <>;
        ingenic,gpio-sleep-low          = <>;
        ingenic,gpio-sleep-high         = <>;
};

&gpb {
        ingenic,gpio-sleep-pullup       = <>;
        ingenic,gpio-sleep-pulldown     = <>;
        ingenic,gpio-sleep-hiz          = <>;
        ingenic,gpio-sleep-low          = <>;
        ingenic,gpio-sleep-high         = <>;
};

&gpc {
        ingenic,gpio-sleep-pullup       = <>;
        ingenic,gpio-sleep-pulldown     = <>;
        ingenic,gpio-sleep-hiz          = <>;
        ingenic,gpio-sleep-low          = <>;
        ingenic,gpio-sleep-high         = <>;
};

&gpd {
        ingenic,gpio-sleep-pullup       = <>;
```

```
        ingenic,gpio-sleep-pulldown     = <>;
        ingenic,gpio-sleep-hiz          = <>;
        ingenic,gpio-sleep-low          = <>;
        ingenic,gpio-sleep-high         = <>;
};
```

22.3.2.4 配置 PA 组电压

用户可根据实际需求将 PA 组电压配置为 3.3V：

```
&pinctrl {
        ingenic,gpa_voltage = <GPIO_VOLTAGE_3V3>;
};
```

注意：当 PA 组使用 3.3V 电压时需要修改核心板 VIDDIO33_CIM 的供电。

22.4 内核编译配置

内核配置 GPIOLIB，配置说明如下：

```
Symbol: GPIOLIB [=y]
Type  : boolean
Prompt: GPIO Support
  Location:
    -> Device Drivers
  Defined at drivers/gpio/Kconfig:34
  Depends on: ARCH_WANT_OPTIONAL_GPIOLIB [=n] || ARCH_REQUIRE_GPIOLIB [=y]
  Selected by: BCM47XX [=n] && <choice> || ARCH_REQUIRE_GPIOLIB [=y] || PINCTRL_AT91 [=n] && PINCTRL [=y]
&& OF [=y] && ARCH_AT91 || PINCTRL_AT91PIO4 [=n] && PINCTRL [=y] && OF [=y] && ARCH_AT91

Symbol: ARCH_REQUIRE_GPIOLIB [=y]
Type  : boolean
  Defined at drivers/gpio/Kconfig:23
  Selects: GPIOLIB [=y]
  Selected by: GPIO_TXX9 [=n] || MIPS_ALCHEMY [=n] && <choice> || AR7 [=n] && <choice> || ATH79 [=n] && <choice>
|| BCM63XX [=n] && <choice> || MACH_INGENIC [=n] && <choice> || MACH_XBURST [=n] &

Symbol: ARCH_WANT_OPTIONAL_GPIOLIB [=n]
Type  : boolean
  Defined at drivers/gpio/Kconfig:13
  Selected by: BCM47XX [=n] && <choice>

Symbol: GPIOLIB_IRQCHIP [=n]
Type  : boolean
  Defined at drivers/gpio/Kconfig:59
  Depends on: GPIOLIB [=y]
  Selects: IRQ_DOMAIN [=y]
```

```
Selected by: PINCTRL_AT91 [=n] && PINCTRL [=y] && OF [=y] && ARCH_AT91 || PINCTRL_AT91PIO4 [=n] && PINCTRL [=y] && OF [=y] && ARCH_AT91 || PINCTRL_AMD [=n] && PINCTRL [=y] && GPIOLIB [=y] || PI
```

22.4.1 内核默认编译配置

内核默认配置 GPIO 驱动，配置界面如下：

```
-*- GPIO Support
[ ]   Debug GPIO calls
[*]   /sys/class/gpio/... (sysfs interface)
      Memory mapped GPIO drivers  --->
      I2C GPIO expanders  --->
      MFD GPIO expanders  ----
      SPI or I2C GPIO expanders  --->
      USB GPIO expanders  ----
```

22.4.2 内核自定义编译配置

用户可根据实际需求自定义 GPIO 编译配置。

22.5 设备节点生成

在内核导出 gpio 节点的前提下， 可以操作/sys/class/gpio 节点， 控制 gpio 输入输出，节点路径：

```
/sys/class/gpio/
```

22.5.1 GPIO 导入导出说明

"export"：用户空间可以通过写其编号到这个文件， 要求内核导出，一个 GPIO 的控制到用户空间。例如：如果内核代码没有申请 GPIO 19,

```
echo 19 > export
```

将会为 GPIO \#19 创建一个 "gpio19" 节点。

"unexport"：导出到用户空间的逆操作。例如

```
echo 19 > unexport
```

将会移除使用"export"文件导出的"gpio19" 节点。

22.5.2 GPIO 属性

GPIO 信号的路径（以 GPIO 42 为例）：

```
/sys/class/gpio/gpio42
```

并有如下的读/写属性：

```
/sys/class/gpio/gpio42/direction
/sys/class/gpio/gpio42/value
/sys/class/gpio/gpio42/edge
/sys/class/gpio/gpio42/active_low
```

- direction：

读取得到 "in" 或 "out"。 这个值通常运行写入。

写入"out" 时,其引脚的默认输出为低电平。 为了确保无故障运行，"low" 或 "high" 的电平值应该写入 GPIO 的配置， 作为初始输出值。

注意:如果内核不支持改变 GPIO 的方向， 或者在导出时内核代码没有明确允许用户空间可以重新配置 GPIO 方向， 那么这个属性将不存在。

- value：

读取得到 0（低电平）或 1（高电平）。 如果 GPIO 配置为输出，这个值允许写操作。任何非零值都以高电平看待。

如果引脚可以配置为中断信号， 且如果已经配置了产生中断的模式（见"edge"的描述）， 你可以对这个文件使用轮询操作(poll(2))，且轮询操作会在任何中断触发时返回。 如果你使用轮询操作(poll(2))，请在 events 中设置 POLLPRI 和 POLLERR。 如果你使用轮询操作(select(2))， 请在 exceptfds 设置你期望的文件描述符。 在轮询操作(poll(2))返回之后， 既可以通过 lseek(2)操作读取 sysfs 文件的开始部分， 也可以关闭这个文件并重新打开它来读取数据。

- edge：

读取得到“none”、“rising”、“falling” 或者“both”，将这些字符串写入这个文件可以选择沿触发模式， 会使得轮询操作(select(2))在"value"文件中返回。这个文件仅有在这个引脚可以配置为可产生中断输入引脚时，才存在。

- active_low：

读取得到 0（假）或 1（真）。 写入任何非零值可以翻转这个属性的\(读写\)值。 已存在或之后通过"edge"属性设置了"rising"。

22.6 应用程序使用说明

23 SMB I2C 接口

23.1 模块功能介绍

SMB 总线是两线串行接口，由串行数据线（SDA）和串行时钟（SCL）组成，这些电线在连接到总线的设备之间传送信息。每个设备都有一个唯一的地址，并且可以根据设备的功能充当“发送器”或“接收器”，在执行数据传输时，设备也可以视为主机或从机。主设备是初始化/终止总线上的数据传输并生成时钟信号以允许该传输的设备。在这段时间内，任何寻址的设备都被视为从设备，SMB 控制器是软件控制的，它充当主机或从机，但是，不支持同时作为主机和从机运行。

i2c 控制器为 SMB，SMB 支持的接口有 i2c,支持 100 Kb/s 和 400 Kb/s，I2C 接口可连接至 pmu, camera，通过 i2c 接口进行配置。

23.2 驱动位置

驱动源码位于

```
drivers/i2c/busses/i2c-ingenic.c
```

23.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

i2c 控制器描述：

```
i2c0: i2c@0x10050000 {
        compatible = "ingenic,x2000-i2c";
        reg = <0x10050000 0x1000>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_I2C0>;
        #address-cells = <1>;
        #size-cells = <0>;
        status = "okay";
};

i2c1: i2c@0x10051000 {
        ……
};

i2c2: i2c@0x10052000 {
        ……;
};

i2c3: i2c@0x10053000 {
```

```
            ......
};

i2c4: i2c@0x10054000 {
            ......
};

i2c5: i2c@0x10055000 {
            ......
};
```

23.3.1 设备树默认配置

设备树默认编译会产生 i2c-adapter0 设备和 i2c-adapter3 设备

23.3.2 设备树自定义配置

用户可根据实际需求打开或关闭某个 i2c adapter 设备，并在设备树的 i2c-adapter 节点下挂接相应的 i2c-client 设备。例如：

```
&i2c3 {
        status = "okay";
        clock-frequency = <100000>;
        timeout = <1000>;
        pinctrl-names = "default";
        pinctrl-0 = <&i2c3_pa>;

        ov2735_0:ov2735@0x3d {
                status = "ok";
                compatible = "ovti,ov2735b";
                reg = <0x3d>;
                pinctrl-names = "default", "default";
                pinctrl-0 = <&vic_pa_low_10bit>;
                pinctrl-1 = <&cim_vic_mclk_pe>;

                ingenic,rst-gpio = <&gpa 10 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,ircutp-gpio = <&gpb 3 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
                ingenic,ircutn-gpio = <&gpb 0 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;

                port {
                        ov2735_ep0:endpoint {
                                remote-endpoint = <&isp0_ep>;
                                bus-width = <10>;       /* Used data lines */
                                data-shift = <0>;       /* Lines 9:0 are used */

                                /* If hsync-active/vsync-active are missing,
```

```
                                embedded BT.656 sync is used */
                                hsync-active = <1>;     /* Active high */
                                vsync-active = <1>;     /* Active high */
                                data-active = <1>;      /* Active high */
                                pclk-sample = <1>;      /* Rising */
                        };
                };
        };
};
```

23.4 内核编译配置

内核配置 I2C_INGENIC，配置说明如下：

```
Symbol: I2C_INGENIC [=y]
Type  : boolean
Prompt: Ingenic SoC based on Xburst arch's I2C controler Driver support
  Location:
    -> Device Drivers
      -> I2C support
        -> I2C support (I2C [=y])
          -> I2C Hardware Bus support
  Defined at drivers/i2c/busses/Kconfig:996
  Depends on: I2C [=y] && HAS_IOMEM [=y]
```

23.4.1 内核默认编译配置

内核默认配置 I2C 控制器驱动，配置界面如下：

```
    *** I2C system bus drivers (mostly embedded / system-on-chip) ***
< > CBUS I2C driver
< > Synopsys DesignWare Platform
< > EMMA Mobile series I2C adapter
< > GPIO-based bitbanging I2C
< > Imagination Technologies I2C SCB Controller
< > OpenCores I2C Controller
< > PCA9564/PCA9665 as platform device
< > Rockchip RK3xxx I2C adapter
< > Simtec Generic I2C interface
< > Xilinx I2C Controller
[*] Ingenic SoC based on Xburst arch's I2C controler Driver support
[ ]   controler i2c no restart mode
(64)  INGENIC I2C Controller FIFO length
[ ]   enable or disable Ingenic Soc's I2C driver debug info
    *** External I2C/SMBus adapter drivers ***
< > Diolan U2C-12 USB adapter
< > Parallel port adapter (light)
< > RobotFuzz Open Source InterFace USB adapter
< > TAOS evaluation module
< > Tiny-USB adapter
    *** Other I2C/SMBus bus drivers ***
```

23.4.2 内核自定义编译配置

1.用户可根据实际需求去掉 I2C 控制器的驱动。

2.可将设备节点导出到用户空间的/dev 下，配置 I2C_CHARDEV，配置说明如下：

```
Symbol: I2C_CHARDEV [=y]
Type  : tristate
Prompt: I2C device interface
  Location:
    -> Device Drivers
      -> I2C support
        -> I2C support (I2C [=y])
  Defined at drivers/i2c/Kconfig:49
  Depends on: I2C [=y]
```

配置界面如下：

```
--- I2C support
[*]   Enable compatibility bits for old user-space
<*>   I2C device interface
<*>   I2C bus multiplexing support
        Multiplexer I2C Chip support  --->
[*]   Autoselect pertinent helper modules
      I2C Hardware Bus support  --->
< >   I2C/SMBus Test Stub
[ ]   I2C slave support
[ ]   I2C Core debugging messages
[ ]   I2C Algorithm debugging messages
[ ]   I2C Bus debugging messages
```

23.5 设备节点生成

打开 I2C_CHARDEV 选项后将在/dev 下生成相应的节点：

```
ls /dev/i2c-
i2c-0  i2c-2  i2c-3  i2c-4
```

23.6 应用程序使用说明

查看 I2C 总线

```
# i2cdetect -l
i2c-0   i2c             i2c0                                    I2C adapter
i2c-2   i2c             i2c2                                    I2C adapter
i2c-3   i2c             i2c3                                    I2C adapter
i2c-4   i2c             i2c4                                    I2C adapter
```

24SSI SPI 接口

24.1 模块功能介绍

SSI 是一个全双工同步串行接口，可以连接到多种外部模拟-数字（A/D）转换器、音频和电信编解码器以及其他使用串行传输数据的协议。X2000 和 M300 支持 SSI 摩托罗拉的串行外设接口（SPI）协议。

- GPIO 功能描述

Name	I/O	Description
SSI_CLK	Output	Serialbit-rateclock
SSI_CE0	Output	Firstslaveselectenable
SSI_DT	Output	Transmitdata(serialdataout)
SSI_DR	Input	Receivedata(serialdatain)

- GPIO 接口

```
SSI0 PB28-31 FUNCTION1
SSI1 PC09-12 FUNCTION2
SSI0 PD08-10 PD13 FUNCTION1
SSI1 PD17-19 PD22 FUNCTION2
```

24.2 驱动源码位置

驱动源码所在位置：

```
kernel/drivers/spi
├── ingenic_spi.c
├── ingenic_spi.h
├── spidev.c
├── spi.c
└── spi-bitbang.c
```

24.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

SPI 控制器描述：

```
spi0: spi@0x10043000 {
                        compatible = "ingenic,spi";
                        reg = <0x10043000 0x1000>;
                        interrupt-parent = <&core_intc>;
                        interrupts = <IRQ_SSI0>;
                        dmas = <&pdma INGENIC_DMA_TYPE(INGENIC_DMA_REQ_SSI0_TX)>,
                                <&pdma INGENIC_DMA_TYPE(INGENIC_DMA_REQ_SSI0_RX)>;
                        dma-names = "tx", "rx";
```

```
                        #address-cells = <1>;
                        #size-cells = <0>;
                        status = "disabled";
 };

spi1: spi1@0x10044000 {
                        ……
};
```

24.3.1 设备树默认配置

设备树默认关闭 SPI 控制器设备。

24.3.2 设备树自定义配置

用户可根据实际需求打开 SPI 控制器设备，例如：

```
&spi0 {
        status = "ok";
        pinctrl-names = "default";
        pinctrl-0 = <&spi0_pb>;

        spi-max-frequency = <54000000>;
        num-cs = <2>;
        cs-gpios = <0>, <0>;
        /*cs-gpios  =  <&gpa  27  GPIO_ACTIVE_HIGH  INGENIC_GPIO_NOBIAS>,  <&gpa  27  GPIO_ACTIVE_HIGH
INGENIC_GPIO_NOBIAS>;*/
        ingenic,chnl = <0>;
        ingenic,allow_cs_same = <1>;
        ingenic,bus_num = <0>;
        ingenic,has_dma_support = <0>; /*选择 dma，需要配置 pdma 通道*/
        ingenic,spi-src-clk = <1>;/*0.ext; 1.ssi*/

        /* Add SPI interface device */
        spidev: spidev@0 {
                compatible = "rohm,dh2228fv";
                reg = <0>;
                spi-max-frequency = <10000000>;
        };
};
```

24.3.3 用 gpio 模拟 spi 协议

为满足一些特殊的协议要求，也可以采用基于 bitbang 的 gpio 模拟 spi 功能.

将如下 spi-gpio 节点，添加设备树根节点下：

例: arch/mips/boot/dts/ingenic/halley5_v20.dts*

```
/ {
```

```
        spi_gpio {
                status = "okay";
                compatible = "spi-gpio";
                #address-cells = <0x1>;
                ranges;
                gpio-sck = <&gpb 28 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                gpio-miso = <&gpb 29 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                gpio-mosi = <&gpb 30 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                cs-gpios = <&gpb 31 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                num-chipselects = <1>;
                /* clients */
                spidev1: spidev1@0 {
                        compatible = "rohm,dh2228fv";
                        reg = <0>;
                        spi-max-frequency = <500000>;
                };
        };
}
```

24.4 内核编译配置

内核配置 INGENIC_SPI，配置说明如下：

```
Symbol: INGENIC_SPI [=n]
Type  : tristate
Prompt: Ingenic SPI Controller
  Location:
        -> Device Drivers
          -> SPI support (SPI [=y])
  Defined at drivers/spi/Kconfig:56
  Depends on: SPI [=y] && SPI_MASTER [=y] && (MACH_XBURST [=n] || MACH_XBURST2 [=y])
  Selects: SPI_BITBANG [=y]
```

24.4.1 内核默认编译配置

内核默认关闭 SPI 控制器驱动

24.4.2 内核自定义编译配置

用户可根据实际需求打开 SPI 控制器编译，配置界面如下：

```
--- SPI support
[ ]   Debug support for SPI drivers (NEW)
      *** SPI Master Controller Drivers ***
<*>   Ingenic SPI Controller
[ ]     Ingenic SoC SSI controller 0 for SPI Host driver (NEW)
[ ]     Use GPIO CE on Ingenic SSI controller 0 (NEW)
< >   Altera SPI Controller (NEW)
-*-   Utilities for Bitbanging SPI masters
< >   Cadence SPI controller (NEW)
< >   GPIO-based bitbanging SPI Master (NEW)
< >   IMG SPFI controller (NEW)
< >   Freescale SPI controller and Aeroflex Gaisler GRLIB SPI controller (NEW)
< >   OpenCores tiny SPI (NEW)
< >   Rockchip SPI controller driver (NEW)
< >   NXP SC18IS602/602B/603 I2C to SPI bridge (NEW)
< >   Analog Devices AD-FMCOMMS1-EBZ SPI-I2C-bridge driver (NEW)
< >   Xilinx SPI controller common module (NEW)
< >   Xilinx ZynqMP GQSPI controller (NEW)
< >   DesignWare SPI controller core support (NEW)
      *** SPI Protocol Masters ***
<*>   User mode SPI device driver support
< >   Infineon TLE62X0 (for power switching) (NEW)
```

24.4.3 用 gpio 模拟 spi 协议

为满足一些特殊的协议要求，也可以采用基于 bitbang 的 gpio 模拟 spi 功能.

```
Device Drivers  --->
    [*] SPI support  --->
        < >  Ingenic SPI Controller    /*去掉 ingenic ssi 控制器配置*/
        -*-  Utilities for Bitbanging SPI masters
        <*>  GPIO-based bitbanging SPI Master
        <*>  User mode SPI device driver support   /*向用户提供设备设备节点*/
```

24.5 设备节点生成

驱动加载成功 log

```
[    0.298196] INGENIC SSI Controller for SPI channel 0 driver register
```

生成设备节点:

```
/dev/spidev0.0
```

当使用 gpio 模拟 spi 协议时，产生设备节点 spidev%d.%d*，例如:

```
/dev/spidev32766.0
```

24.6 应用程序使用说明

内核的 spi 驱动程序是基于 spi 子系统架构编写的

- 应用程序可以使用 spi_demo 进行测试
- 可以在内核空间通过 spi 驱动操作 spi 接口
- 可以在用户空间通过 spi_ioc_transfer 操作 spi 接口

24.6.1 测试 spi 读写

24.6.1.1 源码位置

```
kernel/Documentation/spi/spidev_test.c
```

24.6.1.2 测试方法

1.将 SSI0_DT 和 SSI0_DR 引脚短接

```
# ./spi_test --help
./spi_test: unrecognized option '--help'
Usage: ./spi_test [-DsbdlHOLC3]
  -D --device   device to use (default /dev/spidev1.1)
  -s --speed    max speed (Hz)
  -d --delay    delay (usec)
  -b --bpw      bits per word
  -l --loop     loopback
  -H --cpha     clock phase
  -O --cpol     clock polarity
  -L --lsb      least significant bit first
  -C --cs-high  chip select active high
  -3 --3wire    SI/SO signals shared
  -v --verbose  Verbose (show tx buffer)
  -p            Send data (e.g. "1234\xde\xad")
  -N --no-cs    no chip select
  -R --ready    slave pulls low to pause
  -2 --dual     dual transfer
  -4 --quad     quad transfer
```

2. spidev_test-D/dev/spidev0.0

24.6.1.3 测试结果

```
# ./spi_test -D /dev/spidev0.0
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)
RX | FF FF FF FF FF FF 40 00 00 00 00 95 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF F0 0D
|......@.....................
```

24.6.2 测试 spi nor flash

针对 spi 接口的 nor flash，内核已经有对应驱动支持，如果想使用只需要正确配置即可。

24.6.2.1 源码位置

```
packages/example/Sample/spi/
```

24.6.2.2 设备树配置

设备树位置：

```
x2000/kernel-4.4.94/arch/mips/boot/dts/ingenic/halley5_v20.dts
```

需要在 spi 节点内添加 nor flash 的节点：

```
&spi0 {
	status = "okay";
	......

	/* Add SPI nor flash */
	spinor: spinor@0 {
		status = "okay";
		compatible = "jedec,spi-nor";
		reg = <0>;
		spi-max-frequency = <10000000>;
	};
};
```

24.6.2.3 内核编译配置

1. 编译 kernel-4.4.94/drivers/mtd/spi-nor/spi-nor.c

位置：Device Drivers > Memory Technology Device (MTD) support

选中 SPI-NOR device support，如下图所示：

```
< >   SmartMedia/xD new translation layer
< >   Log panic/oops to an MTD buffer
[ ]   Retain master device when partitioned
      RAM/ROM/Flash chip drivers  --->
      Mapping drivers for chip access  --->
      Self-contained MTD device drivers  --->
[ ]   NAND ECC Smart Media byte order
-*-   NAND Device Support  --->
< >   OneNAND Device Support  ----
      LPDDR & LPDDR2 PCM memory drivers  --->
<*>   SPI-NOR device support  --->
<*>   Enable UBI - Unsorted block images  --->
```

2. 编译 kernel-4.4.94/drivers/mtd/devices/m25p80.c

位置：Device Drivers > Memory Technology Device (MTD) support > Self-contained MTD device drivers

选中 Support most SPI Flash chips (AT26DF, M25P, W25X, ...)，如下图所示：

```
    < > Support for AT45xxx DataFlash (NEW)
    <*> Support most SPI Flash chips (AT26DF, M25P, W25X, ...)
    < > Support SST25L (non JEDEC) SPI Flash chips (NEW)
    < > Uncached system RAM
    < > Physical system RAM
    < > Test driver using RAM
    < > MTD using block device
        *** Disk-On-Chip Device Drivers ***
    < > M-Systems Disk-On-Chip G3
    <*> Ingenic series SFC driver
          Select Ingenic series SFC driver version (Use ingenic sfc d
          the SFC external memory (nor or nand) (Support ingenic sfc-
```

24.6.2.4 测试

1. 确保 nor flash 硬件已经连接好；
2. 在 kernel-4.4.94/drivers/mtd/spi-nor/spi-nor.c 中根据 nor flash 的型号，查找下表看是否有对应的型号支持，如果没有需要配置对应的参数：

```
static const struct flash_info spi_nor_ids[] = {
    /* Atmel -- some are (confusingly) marketed as "DataFlash" */
    { "at25fs010",  INFO(0x1f6601,  0,  32 * 1024,   4,  SECT_4K) },
    { "at25fs040",  INFO(0x1f6604,  0,  64 * 1024,   8,  SECT_4K) },
    ...........
    /* GigaDevice */
    { "gd25q32",  INFO(0xc84016,  0,  64 * 1024,  64,  SECT_4K) },
    { "gd25q64",  INFO(0xc84017,  0,  64 * 1024,  128,  SECT_4K) },
    { "gd25q128",  INFO(0xc84018,  0,  64 * 1024,  256,  SECT_4K) },
    ...........
    { },
};
```

3. 重新编译 kernel 并烧录；
4. 看到如下信息则表示添加成功：

```
[    1.571673] m25p80 spi0.0: xm25qh128b (16384 Kbytes)
```

这里示范的 nor flash 型号为 xm25qh128b，大小为 16MB，具体打印信息与 spi-nor.c 文件中 struct flash_info spi_nor_ids[] 结构体中配置的参数相同；

5. 进入到根文件系统后，可执行如下命令进一步确认，如果增加了一个新的 mtd 分区，则表示添加成功，这里 mtd4 为新添加的 nor flash 对应的分区

```
# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00100000 00020000 "uboot"
mtd1: 00800000 00020000 "kernel"
mtd2: 06000000 00020000 "rootfs"
mtd3: 09700000 00020000 "userdata"
mtd4: 01000000 00010000 "spi0.0"
```

同时，在/dev 目录下也会生成对应的设备节点，这里示范的为 mtd4、mtd4ro、mtdblock4 三个

设备节点，其中 mtd4 和 mtd4ro 为字符设备节点，差别在于 mtd4 为可读写，mtd4ro 只读，mtdblock4 为块设备节点，可读写，如下图所示：

```
# ls /dev/mtd
mtd0        mtd1ro      mtd3        mtd4ro      mtdblock2
mtd0ro      mtd2        mtd3ro      mtdblock0   mtdblock3
mtd1        mtd2ro      mtd4        mtdblock1   mtdblock4
```

6. 接下来可以使用诸如 flash_erase、mtd_debug 等命令对 nor flash 生成的字符设备节点 mtd4 进行读写，同时也可以对块设备节点 mtdblock4 进行格式化、挂载、文件系统读写等一系列操作，这里不再进行示范。

25UART 串口

25.1 模块功能介绍

通用异步接收器/发送器（UART）串行端口。 共有十个 UART：所有 UART 使用相同的编程模型。 每个串行端口都可以在基于中断的模式或基于 DMA 的模式下运行。通用异步接收器/发送器(UART)与 16550 行业标准兼容，可以用作符合红外数据协会（IrDA）串行红外规范 1.1 的慢速红外异步接口。

25.2 驱动位置

驱动源码位置：

```
drivers/tty/serial/ingenic_uart.h
drivers/tty/serial/ingenic_uart.c
```

25.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

UART 控制器描述：

```
uart0: serial@0x10030000 {
        compatible = "ingenic,8250-uart";
        reg = <0x10030000 0x100>;
        reg-shift = <2>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_UART0>;
        clocks = <&extclk 0>;
        clock-names = "uart0";
};
uart1: serial@0x10031000 {
        ……
};
        ……
uart9: serial@0x10039000 {
        ……
};
```

25.3.1 设备树默认配置

在板级设备树 halley5_v20.dts 中，默认编译会产生 uart0，uart3 设备，描述如下：

```
&uart0 {
        status = "okay";
        pinctrl-names = "default";
```

```
        pinctrl-0 = <&uart0_pd>;
};

&uart2 {
        status = "disable";
};

&uart3 {
        status = "okay";
        pinctrl-names = "default";
        pinctrl-0 = <&uart3_pc>;
};
```

25.3.2 设备树自定义配置

用户可根据实际需求打开或关闭某个串口设备。

25.4 内核编译配置

内核配置 SERIAL_INGENIC_UART，配置说明如下：

```
Symbol: SERIAL_INGENIC_UART [=y]
Type  : tristate
Prompt: ingenic serial port support
  Location:
    -> Device Drivers
      -> Character devices
        -> Serial drivers
  Defined at drivers/tty/serial/Kconfig:1632
  Depends on: TTY [=y] && HAS_IOMEM [=y]
  Selects: SERIAL_CORE [=y]
```

25.4.1 内核默认编译配置

内核默认配置 UART 驱动，配置界面如下：

```
< > 8250/16550 and compatible serial support
    *** Non-8250 serial port support ***
< > Xilinx uartlite serial port support
< > SCCNXP serial port support
< > SC16IS7xx serial support
< > Broadcom BCM63xx/BCM33xx UART support
< > Altera JTAG UART support
< > Altera UART support
< > Cadence (Xilinx Zynq) UART support
< > ARC UART driver support
< > Freescale lpuart serial port support
< > Conexant Digicolor CX92xxx USART serial port support
<*> ingenic serial port support
[*]   Console on ingenic soc and compatible serial port
[*]   ingenic baudrate add support greater than 1M
[*]   ingenic uart enable Magic SysRq key
```

25.4.2 内核自定义编译配置

用户可根据实际需求去掉该驱动的配置。

25.5 设备节点生成

驱动加载成功后生成相应的设备节点

```
# ls /dev/ttyS
ttyS1  ttyS2  ttyS3
```

25.6 应用程序使用说明

测试方法

```
# cat /dev/ttyS2 &
# echo "this is serial test string" > /dev/ttyS2
this is serial test string
```

26 eMMC/SD/SDIO 接口

26.1 模块功能介绍

DWC_MSHC 是一种可高度配置和可编程的高性能移动存储主控制器。DWC_MSHC 其数据传输的总接口为 AXI。EMMC 是英文 Embedded Multi-Media Card（嵌入式多媒体卡）的缩写，X2000 支持 EMMC 协议 Electrical Standard（5.1）。

X2000 和 M300 使用的 ap6256 芯片。 AP6256 是基于 BCM4345C5 方案的集成 wifi 和 bluetooth 的功能模块，它包括用于 WiFi 的 SDIO 接口，和用于蓝牙的 UART/PCM 接口。

26.1.1 GPIO 功能描述:

26.1.1.1 EMMC

Name	I/O	Funtion
MSC0_CLK	GPIO-PD17	FUNTION0
MSC0_CMD	GPIO-PD18	FUNTION0
MSC0_DATA	GPIO-(PD19~PD26)	FUNTION0

26.1.1.2 SDIO

Name	I/O	Funtion
SDIO	GPIO-(PD08~PD13)	FUNTION0
WL_REG_ON	GPIO-PD19	
WL_WAKE_HOST	GPIO-PD20	

26.1.1.3 SD

Name	I/O	Funtion
MSC0_CLK	GPIO-PE00	FUNTION0
MSC0_CMD	GPIO-PE01	FUNTION0
MSC0_DATA	GPIO-(PE02~PE05)	FUNTION0

26.1.2 MSC 控制器命名对应关系：

控制器	Base	bootrom	kernel	pm-spec	hardware-pcd
控制器 0	13450000	msc0	msc0	msc0	msc0
控制器 1	13460000	msc1	msc1	sdio	sdio
控制器 2	13490000	msc4	msc2	msc1	msc1

26.2 驱动源码位置

驱动源码位于：

```
drivers/mmc/host/
├——sdhci.c
├——sdhci-ingenic.c
├——ingenic_sdio.c
```

26.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

msc0 控制器描述：

```
msc0: msc@0x13450000 {
                        compatible = "ingenic,sdhci";
                        reg = <0x13450000 0x10000>;
                        status = "disabled";
                        interrupt-parent = <&core_intc>;
                        interrupts = <IRQ_MSC0>;
};
```

msc1 控制器描述：

```
msc1: msc@0x13460000 {
                        compatible = "ingenic,sdhci";
                        reg = <0x13460000 0x10000>;
                        status = "disabled";
                        interrupt-parent = <&core_intc>;
                        interrupts = <IRQ_MSC1>;
};
```

msc2 控制器描述：

```
msc2: msc@0x13490000 {
                        compatible = "ingenic,sdhci";
                        reg = <0x13490000 0x10000>;
                        status = "disabled";
                        interrupt-parent = <&core_intc>;
                        interrupts = <IRQ_MSC2>;
};
```

26.3.1 设备树默认配置

在板级设备树 halley5_v20.dts 中，默认配置产生 msc1 msc2 设备，描述如下：

26.3.1.1 EMMC

```
&msc0 {
        status = "disable";
        /*mmc-hs200-1_8v;*/
        cap-mmc-highspeed;
        non-removable;
        max-frequency = <50000000>;
        bus-width = <4>;
        non-removable;
        voltage-ranges = <1800 3300>;

        /* special property */
        ingenic,wp-gpios = <0>;
        ingenic,cd-gpios = <0>;
        ingenic,rst-gpios = <0>;
};
```

26.3.1.2 SDIO

```
&msc1 {
        status = "okay";
        pinctrl-names ="default","enable", "disable";
        pinctrl-0 = <&msc1_4bit>;
        pinctrl-1 = <&rtc32k_enable>;
        pinctrl-2 = <&rtc32k_disable>;

        sd-uhs-sdr104;
        max-frequency = <100000000>;
        bus-width = <4>;
        voltage-ranges = <1800 3300>;
        non-removable;

        ingenic,sdio_clk = <1>;
        keep-power-in-suspend;

        /* special property */
        ingenic,sdr-gpios = <0>;
        ingenic,wp-gpios = <0>;
        ingenic,cd-gpios = <0>;
        ingenic,rst-gpios = <0>;
        ingenic,removal-manual;     /*removal-dontcare,     removal-nonremovable,     removal-removable,
removal-manual*/

        bcmdhd_wlan: bcmdhd_wlan {
```

```
                compatible = "android,bcmdhd_wlan";
                ingenic,sdio-irq = <&gpd 0 IRQ_TYPE_LEVEL_HIGH INGENIC_GPIO_NOBIAS>;
                ingenic,sdio-reset = <&gpd 1 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
        };
};
```

26.3.1.3 SD

```
&msc2 {
        status = "okay";
        pinctrl-names ="default";
        pinctrl-0 = <&msc2_4bit>;
        sd-uhs-sdr104;
        /*cap-mmc-highspeed;*/
        max-frequency = <200000000>;
        cd-inverted;
        bus-width = <4>;
        voltage-ranges = <1800 3300>;

        /* special property */
        ingenic,sdr-gpios = <&gpc 0 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
        ingenic,wp-gpios = <0>;
        ingenic,cd-gpios = <&gpc 12 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
        ingenic,rst-gpios = <0>;
};
```

26.3.2 设备树自定义配置

26.3.2.1 EMMC

属性名称	说明
● mmc-hs200-1_8v	配置 emmc 支持 hs200 传输模式
● cap-mmc-highspeed	配置 emmc 支持 highspeed 传输模式
● non-removable	配置 emmc 属性为不可移动卡
● max-frequency	通过配置该参数来指定当前模式下最大频率
● enable_cpm_rx_tuning	该选项用于设置手动调试 MSC rx 相位，默认控制器自动 tuning
● enable_cpm_rx_tuning	选项用于设置手动调试 MSC tx 相位
● bus-width	配置数据总线宽度，支持 1bit、4bit、8bit
● voltage-ranges	指定电压范围
● ingenic,wp-gpios	写保护
● ingneic,cd-gpios	卡检查
● ingenic,rst-gpios	硬件 reset

26.3.2.2 SDIO

属性名称	说明
sd-uhs-sdr104	sdr104 模式
max-frequency	通过配置该参数来指定当前模式下最大频率
bus-width	配置数据总线宽度，支持 1bit、4bit、8bit
voltage-ranges	指定电压范围
non-removable	配置 emmc 属性为不可移动卡
ingenic,sdio_clk	选项用于设置手动调试 MSC tx 相位
keep-power-in-suspend	休眠保持供电
ingenic,wp-gpios	写保护
ingneic,cd-gpios	卡检查
ingenic,rst-gpios	硬件 reset
ingenic,removal-manual	removal-dontcare, removal-nonremovable, removal-removable, removal-manual
ingenic,sdio-irq	Wifi 中断
ingenic,sdio-reset	Wifi reset

26.3.2.3 SD

属性名称	说明
sd-uhs-sdr104	sdr104 模式
max-frequency	通过配置该参数来指定当前模式下最大频率
cd-inverted	配置支持 SD 卡热插拔
bus-width	配置数据总线宽度，支持 1bit、4bit、8bit
voltage-ranges	指定电压范围
ingenic,sdr-gpios	通过 GPIO 配置 SD 卡外部电路 3.3V 到 1.8V 电压切换
ingenic,wp-gpios	配置 CD pin 用于 SD 卡检测
ingenic,rst-gpios	硬件 reset

26.4 内核编译配置

内核配置 MMC_ SDHCI_INGENIC，配置说明如下：

```
Symbol: MMC_SDHCI_INGENIC [=y]
Type  : tristate
Prompt: Ingenic(XBurst2)  MMC/SD Card Controller(MSC) support
 Location:
   -> Device Drivers
     -> MMC/SD/SDIO card support (MMC [=y])
 Defined at drivers/mmc/host/Kconfig:7
 Depends on: MMC [=y] && (SOC_X2000 [=n] || SOC_X2000_V12 [=y] || SOC_M300 [=n])
Selects: MMC_SDHCI [=y]
```

26.4.1 内核默认编译配置

26.4.1.1 EMMC/SD/SDIO 基础配置

```
Device Drivers --->
    <*> MMC/SD/SDIO card support --->
        --- MMC/SD/SDIO card support
        [ ] MMC debugging
            *** MMC/SD/SDIO Card Drivers ***
        <*> MMC block device driver
        (16) Number of minors per block device
        [*] Use bounce buffer for simple hosts
        < > SDIO UART/GPS class support
        < > MMC host test driver
            *** MMC/SD/SDIO Host Controller Drivers ***
        <*> Ingenic(XBurst2) MMC/SD Card Controller(MSC) support
        -*- Secure Digital Host Controller Interface support
```

26.4.1.2 SDIO wifi 相关配置

```
Device Drivers
    --- Network device support
    [*] Network core driver support
    [*] Wireless LAN --->
        <*> Broadcom FullMAC wireless cards support
        (/firmware/fw_bcm43456c5_ag.bin) Firmware path
        (/firmware/nvram_ap6256.txt) NVRAM path
            Enable Chip Interface (SDIO bus interface support) --->
            Interrupt type (Out-of-Band Interrupt) --->
```

26.4.2 内核自定义编译配置

内核配置名称	说明
MMC_BLOCK_MINORS	每个块设备的最大分区数
BCMDHD_FW_PATH	配置 wifi 模块固件路径
BCMDHD_NVRAM_PATH	配置 wifi 模块固件路径

26.5 设备节点生成

26.5.1 EMMC

驱动加载成功：

```
[ 1.575803] mmc1: new ultra high speed SDR104 SDHC card at address aaaa
```

```
[ 1.593001] mmcblk0: mmc1:aaaa SC16G 14.8 GiB
[ 1.601920] Alternate GPT is invalid, using primary GPT.
[ 1.607432] mmcblk0: p1 p2 p3 p4 p5 p6 p7
```

产生设备节点：

```
/dev/mmcblk0 /dev/mmcblk0p1~p7 /*对应 emmc 的分区*/
```

26.5.2 SDIO

驱动加载成功：

```
[ 1.124644] Dongle Host Driver, version 1.363.59.144.11 (r) [ 1.124644] Compiled from [ 1.125120] Register
interface [wlan0] MAC: 00:90:4c:11:22:33 [ 1.125120] [ 1.125195] dhd_module_init: Exit err=0
```

26.5.3 SD

驱动加载成功：

```
[ 1.575803] mmc1: new ultra high speed SDR104 SDHC card at address aaaa
[ 1.593001] mmcblk0: mmc1:aaaa SC16G 14.8 GiB
[ 1.601920] Alternate GPT is invalid, using primary GPT.
[ 1.607432] mmcblk0: p1 p2 p3 p4 p5 p6 p7
```

产生设备节点：

```
/dev/mmcblk0/dev/mmcblk0p1~p7  /*对应 sd 的分区*/
```

26.6 应用程序使用说明

26.6.1 EMMC/SD 测试方法

26.6.1.1 写测试

```
# dd if=/dev/zero of=/dev/mmcblk0 bs=1M count=100 conv=fsync
```

26.6.1.2 读测试

```
# sync; echo 3 > proc/sys/vm/drop_caches
#    time dd if=/dev/mmcblk0 of=/dev/null bs=1M count=100
```

26.6.2 SDIO 测试方法

26.6.2.1 配置网络方法

1. /etc/wpa_supplicant

```
# cat /etc/wpa_supplicant.conf
```

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
country=GB"

network={
ssid="Guest" /*连接 WiFi 账户*/
psk="ingenic*123" /*连接 Wifi 密码*/
bssid=
priority=1
}
```

2. 执行 wifi_up.sh

```
# wifi_up.sh
```

```
[ 13.285811] dhd_open: Enter 843c4800
[ 13.290329] dhd_conf_read_config: Ignore config file /firmware/config.txt
[ 13.297356] Final fw_path=/firmware/fw_bcm43456c5_ag.bin
[ 13.302873] Final nv_path=/firmware/nvram_ap6256.txt
[ 13.308002] Final clm_path=/firmware/clm_bcmdhd.blob
[ 13.313152] Final conf_path=/firmware/config.txt
[ 13.317926] dhd_set_bus_params: set use_rxchain 0
[ 13.322804] dhd_set_bus_params: set txglomsize 36
[ 13.328047] dhd_os_open_image: /firmware/fw_bcm43456c5_ag.bin (579388 bytes) open success
[ 13.409630] dhd_os_open_image: /firmware/nvram_ap6256.txt (2440 bytes) open success
[ 13.516230] bcmsdh_oob_intr_register: HW_OOB enabled
[ 13.521376] bcmsdh_oob_intr_register OOB irq=72 flags=0x4
[ 13.556873] Firmware up: op_mode=0x0005, MAC=c0:84:7d:31:c8:c8
[ 13.640263] dhd_txglom_enable: enable 1
[ 13.748733] dhd_open: Exit ret=0
# udhcpc: started, v1.26.2
Successfully initialized wpa_supplicant
[ 16.320669] Connecting with 8c:0c:90:98:47:cd ssid "wifi ssid", len (5) channel=153
[ 16.367807] wl_bss_connect_done succeeded with 8c:0c:90:98:47:cd
[ 16.401524] wl_bss_connect_done succeeded with 8c:0c:90:98:47:cd
udhcpc: sending select for 10.4.30.146
adding dns 219.141.136.10
adding dns 202.106.0.20
```

3. 执行 ping 命令连网

```
# ping www.baidu.com
```

```
PING www.baidu.com (220.181.38.150): 56 data bytes
```

```
64 bytes from 220.181.38.150: seq=0 ttl=53 time=14.347 ms
64 bytes from 220.181.38.150: seq=1 ttl=53 time=9.873 ms
64 bytes from 220.181.38.150: seq=2 ttl=53 time=12.889 ms
```

26.6.2.2 airkiss 配网测试

AirKiss 是微信硬件平台为 Wi-Fi 设备提供的微信配网、局域网发现和局域网通讯的技术。

1. 配网

```
注：以下操作必须在 WIFI(外网)环境下配置
使用 AirkissDubugger 软件（手机端下载）配置 wifi
```

2. 执行 airkiss 命令

```
# airkiss
```

```
[ 113.669340] dhd_open: Enter 842f1000
[ 113.677706] Dongle Host Driver, version 1.579.77.41.26 (r-20200429-2)
[ 113.695902] ======== PULL WL_REG_ON(-1) HIGH! ========
[ 114.122604] F1 signature read @0x18000000=0x15294345
[ 114.130573] F1 signature OK, socitype:0x1 chip:0x4345 rev:0x9 pkg:0x2
[ 114.137706] DHD: dongle ram size is set to 819200(orig 819200) at 0x198000
[ 114.153611] [dhd] dhd_conf_read_config : Ignore config file /firmware/config.txt
[ 114.161413] [dhd] dhd_conf_set_path_params : Final fw_path=/firmware/fw_bcm43456c5_ag.bin
[ 114.169851] [dhd] dhd_conf_set_path_params : Final nv_path=/firmware/nvram_ap6256.txt
[ 114.177960] [dhd] dhd_conf_set_path_params : Final clm_path=/firmware/clm_bcm43456c5_ag.blob
[ 114.186695] [dhd] dhd_conf_set_path_params : Final conf_path=/firmware/config.txt
[ 114.194942] dhd_os_open_image: /firmware/fw_bcm43456c5_ag.bin (579388 bytes) open success
[ 114.266737] dhd_os_open_image: /firmware/nvram_ap6256.txt (2440 bytes) open success
[ 114.283245] dhdsdio_write_vars: Download, Upload and compare of NVRAM succeeded.
[ 114.580570] [dhd-wlan0] wl_android_wifi_on : Success
Easy setup target library v4.0.0
state: 0 --> 1
state: 1 --> 3
state: 3 --> 5
ssid: JZ_SW       /*软件端配置的网络账号*/
password: jz_sw%#!135   /*软件端配置的网络密码*/
/etc/wpa_supplicant.conf create successfully!
random: 0xc1
time elapsed: 0s
# udhcpc: started, v1.31.1
Successfully initialized wpa_supplicant
[ 118.985525] [dhd-wlan0] wl_run_escan : LEGACY_SCAN sync ID: 0, bssidx: 0
wlan0: Trying to associate with c4:01:7c:78:c3:dd (SSID='JZ_SW' freq=5765 MHz)
[ 121.338272] [dhd-wlan0] wl_cfg80211_connect : Connecting with c4:01:7c:78:c3:dd ssid "JZ_SW", len (5),
```

```
sec=wpa2psk/mfpn/tkipaes, channel=153
[ 121.402046] [dhd-wlan0] wl_ext_iapsta_event : [S] Link UP with c4:01:7c:78:c3:dd
[ 121.409697] [dhd-wlan0] wl_notify_connect_status : wl_bss_connect_done succeeded with c4:01:7c:78:c3:dd
wlan0: Associated with c4:01:7c:78:c3:dd
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
wlan0: WPA: Key negotiation completed with c4:01:7c:78:c3:dd [PTK=CCMP GTK=TKIP]
wlan0: CTRL-EV[ 121.451231] [dhd-wlan0] wl_notify_connect_status : wl_bss_connect_done succeeded with
c4:01:7c:78:c3:dd vndr_oui: 00-90-4C 00-13-92
ENT-CONNECTED - Connection to c4:01:7c:78:c3:dd completed [id=0 id_str=]
udhcpc: sending select for 10.10.30.203
udhcpc: lease of 10.10.30.203 obtained, lease time 86400
adding dns 192.168.1.2
```

3. 执行 ping 命令连接网络

```
# ping www.baidu.com
```

```
PING www.baidu.com (220.181.38.150): 56 data bytes
64 bytes from 220.181.38.150: seq=0 ttl=53 time=14.347 ms
64 bytes from 220.181.38.150: seq=1 ttl=53 time=9.873 ms
```

26.6.2.3 AP 模式使用方法

1. AP 模式下的配置文件（/etc/hostapd.conf）

```
interface=wlan0
driver=nl80211
ssid=ingenic
channel=1
hw_mode=g
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
#wpa=2
#wpa_passphrase=12345678
#wpa_key_mgmt=WPA-PSK
#wpa_pairwise=TKIP
#rsn_pairwise=CCMP
```

2. 进入 AP 模式

```
# wifi_ap_mode_start.sh
```

```
wlan0: interface state UNINITIALIZED->ENABLED
```

```
wlan0: AP-ENABLED
```

3. 执行 ifconfig，查看连接信息

```
# ifconfig
```

```
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 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:1
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

wlan0 Link encap:Ethernet HWaddr C0:84:7D:6A:F1:CD
inet addr:192.168.1.1 Bcast:192.168.1.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:46 errors:0 dropped:21 overruns:0 frame:0
TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6509 (6.3 KiB) TX bytes:744 (744.0 B)
```

4. 使用手机（或电脑）连接 wifi

```
连接 SSID 为“ingenic”对应配置文件 ssid=ingenic，密码可以使用 wpa_passphrase = 12345678 指定(本测试未设置密码)
```

连接成功会打印如下信息：

```
[ 515.831824] [dhd-wlan0] wl_ext_iapsta_event : [A] connected device 00:26:c6:58:50:54
[ 515.839828] [dhd-wlan0] wl_notify_connect_status_ap : connected device 00:26:c6:58:50:54
[ 515.851919] [dhd] CFG80211-ERROR) wl_cfg80211_change_station : WLC_SCB_AUTHORIZE sta_flags_mask not set
udhcpd: sending OFFER to 192.168.1.2
udhcpd: sending ACK to 192.168.1.2
```

5. 连接成功,Ping 网络测试

```
# ping 192.168.1.2
```

```
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: seq=0 ttl=64 time=8.148 ms
64 bytes from 192.168.1.2: seq=1 ttl=64 time=15.411 ms
64 bytes from 192.168.1.2: seq=2 ttl=64 time=12.505 ms
64 bytes from 192.168.1.2: seq=3 ttl=64 time=7.654 ms
64 bytes from 192.168.1.2: seq=4 ttl=64 time=7.505 ms
```

27USB OTG 控制器接口

27.1 模块功能介绍

通用串行总线 USB(Universal Serial Bus) 是一种新兴的并逐渐取代其他接口标准的数据通信方式，由 Intel、Compaq、Digital、IBM、Microsoft、NEC 及 Northern Telecom 等计算机公司和通信公司于 1995 年联合制定，并逐渐形成了行业标准。

USB 总线作为一种高速串行总线，其极高的传输速度可以满足高速数据传输的应用环境要求，且该总线还兼有供电简单（可总线供电）、安装配置便捷（支持即插即用和热插拔）、 扩展端口简易（通过集线器最多可扩展 127 个外设）、传输方式多样化（4 种传输模式），以及兼容良好（产品升级后向下兼容）等优点。

USB OTG 控制器实现了 USB 主机和多种可同时访问的便携式外围设备之间进行串行数据交换。

27.2 驱动源码位置

驱动源码所在位置：

```
drivers/usb/dwc2$ tree
├── core.c
├── core.h
├── core_intr.c
├── debugfs.c
├── debug.h
├── gadget.c
├── hcd.c
├── hcd_ddma.c
├── hcd.h
├── hcd_intr.c
├── hcd_queue.c
├── hw.h
├── Kconfig
├── Makefile
├── pci.c
├── platform.c
```

27.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

设备树描述：

```
otg: otg@0x13500000 {
                        compatible = "ingenic,dwc2-hsotg";
                        reg = <0x13500000 0x40000>;
                        interrupt-parent = <&core_intc>;
                        interrupts = <IRQ_OTG>;
                        ingenic,usbphy=<&otg_phy>;
                        status = "disabled";
 };

otg_phy: otg_phy {
                        #address-cells = <1>;
                        #size-cells = <1>;
                        compatible = "ingenic,innophy";
                        reg = <0x10000000 0x100 0x10078000 0x110>;

 };
```

27.3.1 设备树默认配置

设备树默认编译会产生 otg 和 otg_phy 设备，在 halley5_v20.dts 中配置如下：

```
&otg {
        g-use-dma;
        dr_mode = "otg";
        status = "okay";
};

&otg_phy {
      dr_mode = "otg";
      compatible = "ingenic,innophy", "syscon";
      ingenic,id-dete-gpio = <&gpc 27 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
      ingenic,vbus-dete-gpio = <&gpd 17 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
      Ingenic,drvvbus-gpio = <&gpe 22 GPIO_ACTIVE_HIGH INGENIC_GPIO_NOBIAS>;
      status = "okay";
};
```

27.3.2 设备树自定义配置

用户可根据实际需求关闭该节点，或进行以下配置：

属性名称	说明
● ingenic,id-dete-gpio	配置 usb detect id 的 GPIO
● ingenic,vbus-dete-gpio	配置 usb 用于插拔检测的 GPIO
● Ingenic,drvvbus-gpio	配置 usb 用于控制 vbus 的 GPIO

27.4 内核编译配置

需要根据 USB 不同功能选择不同的配置，USB composite 功能可以支持多个 device 功能同时使用。

USB 作为 host:

	支持功能	说明
●	mass storage	作为主机,可识别大容量存储设备
●	usb camera	作为主机,可识别 USB 摄像头设备
●	hid	作为主机,可以识别人机交互设备,如鼠标..

USB 作为 device:

	支持功能	说明
●	mass storage	usb 作为大容量存储设备
●	adb (Android Debug Bridge)	用于 USB debug 调试功能
●	hid	usb 作为人机交互设备,如鼠标、键盘
●	uvc (usb video class)	usb 作为视频类设备
●	uac1.0 (usb audio class1.0)	usb 作为音频类设备
●	printer	usb 作为打印机设备
●	rndis	usb 作为网卡设备
●	serial	usb 作为串口设备

27.4.1 内核默认编译配置

USB host 默认支持 mass storage、usb camera、hid。

USB device composite 默认支持 mass storage、adb、uvc、uac1.0、printer、rndis 可以同时使用。

27.4.2 内核自定义编译配置

27.4.2.1USB host 内核配置

27.4.2.1.1 USB host Mass Storage

```
(1)VFAT (Windows-95) fs support
Symbol: VFAT_FS [=y]
Type : tristate
Prompt: VFAT (Windows-95) fs support
Location:
-> File systems
-> DOS/FAT/NT Filesystems
Defined at fs/fat/Kconfig:60
Depends on: BLOCK [=y]
Selects: FAT_FS [=y]
```

```
(2)Codepage 437 (United States, Canada)
Symbol: NLS_CODEPAGE_437 [=y]
Type : tristate
Prompt: Codepage 437 (United States, Canada)
Location:
-> File systems
-> Native language support (NLS [=y])
Defined at fs/nls/Kconfig:39
Depends on: NLS [=y]

(3)NLS ISO 8859-1 (Latin 1; Western European Languages)
Symbol: NLS_ISO8859_1 [=y]
Type : tristate
Prompt: NLS ISO 8859-1 (Latin 1; Western European Languages)
Location:
-> File systems
-> Native language support (NLS [=y])
Defined at fs/nls/Kconfig:318
Depends on: NLS [=y]

(4)SCSI disk support
Symbol: BLK_DEV_SD [=y]
Type : tristate
Prompt: SCSI disk support
Location:
-> Device Drivers
-> SCSI device support
Defined at drivers/scsi/Kconfig:73
Depends on: SCSI [=y]

(5)Symbol: USB_STORAGE [=y]
Type : tristate
Prompt: USB Mass Storage support
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> Support for Host-side USB (USB [=y])
Defined at drivers/usb/storage/Kconfig:8
Depends on: USB_SUPPORT [=y] && USB [=y] && SCSI [=y]
```

27.4.2.1.2 USB host camera

```
Symbol: USB_VIDEO_CLASS [=y]
Type : tristate
Prompt: USB Video Class (UVC)
Location:
-> Device Drivers
-> Multimedia support (MEDIA_SUPPORT [=y])
-> Media USB Adapters (MEDIA_USB_SUPPORT [=y])
Defined at drivers/media/usb/uvc/Kconfig:1
Depends on: USB [=y] && MEDIA_SUPPORT [=y] && MEDIA_USB_SUPPORT [=y] && MEDIA_CAMERA_SUPPORT [=y] && VIDEO_V4L2
[=y]
Selects: VIDEOBUF2_VMALLOC [=n]
```

27.4.2.1.3 USB host hid mouse

```
(1)USB HIDBP Mouse (simple Boot) support
Symbol: USB_MOUSE [=y]
Type : tristate
Prompt: USB HIDBP Mouse (simple Boot) support
Location:
-> Device Drivers
-> HID support
-> USB HID support
-> USB HID Boot Protocol drivers
Defined at drivers/hid/usbhid/Kconfig:66
Depends on: USB_HID [=n]!=y && EXPERT [=y] && USB [=y] && INPUT [=y]

(2)Mouse interface
Symbol: INPUT_MOUSEDEV [=y]
Type : tristate
Prompt: Mouse interface
Location:
-> Device Drivers
-> Input device support
-> Generic input layer (needed for keyboard, mouse, ...) (INPUT [=y])
Defined at drivers/input/Kconfig:95
Depends on: !UML && INPUT [=y]
```

27.4.2.2 USB gadet 内核配置

27.4.2.2.1 USB device mass storage

```
Symbol: USB_CONFIGFS_MASS_STORAGE [=y]
Type : boolean
Prompt: Mass storage
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:338
Depends on: <choice> && USB_CONFIGFS [=y] && BLOCK [=y]
Selects: USB_F_MASS_STORAGE [=y]
```

27.4.2.2.2 USB device adb

```
Symbol: USB_CONFIGFS_F_FS [=y]
Type : boolean
Prompt: Function filesystem (FunctionFS)
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:362
Depends on: <choice> && USB_CONFIGFS [=y]
Selects: USB_F_FS [=y]
```

27.4.2.2.3 USB device hid

```
Symbol: USB_CONFIGFS_F_HID [=y]
Type : boolean
Prompt: HID function
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:419
Depends on: <choice> && USB_CONFIGFS [=y]
```

```
Selects: USB_F_HID [=n]
```

27.4.2.2.4 USB device uvc

```
(1)USB Video Class (UVC)
Symbol: USB_VIDEO_CLASS [=y]
Type : tristate
Prompt: USB Video Class (UVC)
Location:
-> Device Drivers
-> Multimedia support (MEDIA_SUPPORT [=y])
-> Media USB Adapters (MEDIA_USB_SUPPORT [=y])
Defined at drivers/media/usb/uvc/Kconfig:1
Depends on: USB [=y] && MEDIA_SUPPORT [=y] && MEDIA_USB_SUPPORT [=y] && MEDIA_CAMERA_SUPPORT [=y] && VIDEO_V4L2
[=y]
Selects: VIDEOBUF2_VMALLOC [=n]

(2)USB Webcam function
Symbol: USB_CONFIGFS_F_UVC [=y]
Type : boolean
Prompt: USB Webcam function
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:429
Depends on: <choice> && USB_CONFIGFS [=y] && VIDEO_DEV [=y]
Selects: VIDEOBUF2_VMALLOC [=y] && USB_F_UVC [=n]

(3)Enable usb dwc2 highwidth fifo
Symbol: USB_DWC2_HIGHWIDTH_FIFO [=y]
Type : boolean
Prompt: Enable usb dwc2 highwidth fifo
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> DesignWare USB2 DRD Core Support (USB_DWC2 [=y])
Defined at drivers/usb/dwc2/Kconfig:64
Depends on: USB_SUPPORT [=y] && USB_DWC2 [=y]
```

27.4.2.2.5 USB device remote NDIS

```
Symbol: USB_CONFIGFS_RNDIS [=y]
Type : boolean
Prompt: RNDIS
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:297
Depends on: <choice> && USB_CONFIGFS [=y] && NET [=y]
Selects: USB_U_ETHER [=n] && USB_F_RNDIS [=n]
```

27.4.2.2.6 USB device serial

```
Symbol: USB_CONFIGFS_SERIAL [=y]
Type : boolean
Prompt: Generic serial bulk in/out
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:241
Depends on: <choice> && USB_CONFIGFS [=y] && TTY [=y]
Selects: USB_U_SERIAL [=n] && USB_F_SERIAL [=n]

Symbol: USB_CONFIGFS_ACM [=y]
Type : boolean
Prompt: Abstract Control Model (CDC ACM)
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:250
Depends on: <choice> && USB_CONFIGFS [=y] && TTY [=y]
Selects: USB_U_SERIAL [=n] && USB_F_ACM [=n]
```

```
Symbol: USB_CONFIGFS_OBEX [=y]
Type : boolean
Prompt: Object Exchange Model (CDC OBEX)
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:260
Depends on: <choice> && USB_CONFIGFS [=y] && TTY [=y]
Selects: USB_U_SERIAL [=n] && USB_F_OBEX [=n]
```

27.4.2.2.7 USB device printer

```
Symbol: USB_CONFIGFS_F_PRINTER [=y]
Type : boolean
Prompt: Printer function
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
Defined at drivers/usb/gadget/Kconfig:469
Depends on: <choice> && USB_CONFIGFS [=y]
Selects: USB_F_PRINTER [=y]
```

27.4.2.2.8 USB device uac1.0

```
Symbol: USB_CONFIGFS_F_UAC1 [=y]
Type : boolean
Prompt: Audio Class 1.0
Location:
-> Device Drivers
-> USB support (USB_SUPPORT [=y])
-> USB Gadget Support (USB_GADGET [=y])
-> USB Gadget Drivers (<choice> [=y])
-> USB functions configurable through configfs (USB_CONFIGFS [=y])
```

```
Defined at drivers/usb/gadget/Kconfig:402
Depends on: <choice> && USB_CONFIGFS [=y] && SND [=y]
Selects: USB_LIBCOMPOSITE [=y] && SND_PCM [=y] && USB_F_UAC1 [=y]
```

27.4.2.3 USB legacy 内核配置

27.4.2.3.1 USB device serial

```
Symbol: USB_G_SERIAL [=y]
Type : tristate
Prompt: Serial Gadget (with CDC ACM and CDC OBEX support)
  Location:
    -> Device Drivers
      -> USB support (USB_SUPPORT [=y])
        -> USB Gadget Support (USB_GADGET [=y])
          -> USB Gadget Drivers (<choice> [=y])
  Defined at drivers/usb/gadget/legacy/Kconfig:260
  Depends on: <choice> && TTY [=y]
  Selects: USB_U_SERIAL [=y] && USB_F_ACM [=y] && USB_F_SERIAL [=y] && USB_F_OBEX [=y] && USB_LIBCOMPOSITE
[=y]
```

27.4.2.3.2 USB device uvc

```
Symbol: USB_G_WEBCAM [=y]
Type : tristate
Prompt: USB Webcam Gadget
  Location:
    -> Device Drivers
      -> USB support (USB_SUPPORT [=y])
        -> USB Gadget Support (USB_GADGET [=y])
          -> USB Gadget Drivers (<choice> [=y])
  Defined at drivers/usb/gadget/legacy/Kconfig:471
  Depends on: <choice> && VIDEO_DEV [=y]
  Selects: USB_LIBCOMPOSITE [=y] && VIDEOBUF2_VMALLOC [=y] && USB_F_UVC [=y] && USB_DWC2_HIGHWIDTH_FIFO [=y]
```

27.4.2.4 USB gadget 描述符配置

hid、serial 不支持复合设备，需要手动修改配置。

usb composite 和相关功能描述符配置路径如下：

```
buildroot/package/ingenic/system_config/usb-device
├── S90usb
└── usb
```

```
├── adb
├── hid
├── mass_storage
├── printer
├── rndis
├── serial
├── uac1
├── udc_daemon
└── uvc
```

27.5 设备节点生成

无

27.6 应用程序使用说明

27.6.1 USB host

27.6.1.1 USB host Mass Storage

1. 插入U盘弹出打印信息

```
[ 46.070034] usb 1-1: new high-speed USB device number 2 using dwc2
[ 46.296250] usb 1-1: New USB device found, idVendor=0951, idProduct=1666
[ 46.303202] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 46.310592] usb 1-1: Product: DataTraveler 3.0
[ 46.315185] usb 1-1: Manufacturer: Kingston
[ 46.319508] usb 1-1: SerialNumber: 60A44C413C4EB211A98A00A0
[ 46.325821] usb-storage 1-1:1.0: USB Mass Storage device detected
[ 46.342363] scsi host0: usb-storage 1-1:1.0
[ 47.726378] scsi 0:0:0:0: Direct-Access Kingston DataTraveler 3.0 PMAP PQ: 0 ANSI: 6
[ 47.736215] sd 0:0:0:0: [sda] 30277632 512-byte logical blocks: (15.5 GB/14.4 GiB)
[ 47.752538] sd 0:0:0:0: [sda] Write Protect is off
[ 47.759990] sd 0:0:0:0: [sda] No Caching mode page found
[ 47.765648] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 47.780215] sda: sda1
[ 47.789136] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

2. 挂在U盘到文件系统

```
# mount -t vfat /dev/sda1 mnt/
```

3. 进入/mnt 下,进行数据交互

27.6.1.2 USB host camera

1. 插入摄像头识别成功

```
[   56.560037] usb 1-1: new high-speed USB device number 2 using dwc2
[   56.910300] usb 1-1: New USB device found, idVendor=058f, idProduct=5608
[   56.917231] usb 1-1: New USB device strings: Mfr=3, Product=1, SerialNumber=0
[   56.924635] usb 1-1: Product: USB 2.0 Camera
[   56.929114] usb 1-1: Manufacturer: Alcor Micro, Corp.
[   56.939481] uvcvideo: Found UVC 1.00 device USB 2.0 Camera (058f:5608)
[   56.949251] input: USB 2.0 Camera as /devices/platform/ahb2/13500000.otg/usb1/1-1/1-1:1.0/input/input0
```

2. 拍照测试

```
usage: uvcview [-d <device>] [-c <count>]
--help -H print this message
--print_formats -F print video device info
--device -d , video device, default is /dev/video0
--width -w grab width
--height -h grab height
--count -c set the count to grab
--rate -r frame sample rate(fps)
--yuv -y use yuyv input format
--timeout -t select timeout
--match -m do picture match test
--perf -p do performance test
```

```
注：*/dev/vidio5是usb camera生成的标准video节点。*
# ./grab -w 640 -h 480 -d /dev/video5 -y -c 3
```

3. 生成图像

```
p-0.jpg  p-1.jpg  p-2.jpg
```

27.6.1.3 USB host hid mouse

1. 插入鼠标设备到控制器

```
[ 117.100038] usb 1-1: new low-speed USB device number 2 using dwc2
[ 117.313977] usb 1-1: New USB device found, idVendor=046d, idProduct=c077
[ 117.320929] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 117.328305] usb 1-1: Product: USB Optical Mouse
[ 117.333098] usb 1-1: Manufacturer: Logitech
[ 117.338568] input: Logitech USB Optical Mouse as
/devices/platform/ahb2/13500000.otg/usb1/1-1/1-1:1.0/input/input0
```

2. 执行捕获时间的程序后移动鼠标触发事件

```
# cd /testsuite/usb_test/usb_host/getevent_test/
# ./getevent_test 0
```

```
/dev/input/mouse0
evdev version: 0.0.0
name:
features: relative reserved unknown unknown unknown unknown unknown unknown unknown
/dev/input/mouse0: open, fd = 3
Sun Mar 1 16:38:18 2020.000000 type 0x0011; code 0x000e; value 0x00000000; Led
[ 125.626285] random: nonblocking pool is initialized
Wed Dec 25 06:26:16 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
Thu Dec 26 00:59:52 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
Mon Dec 23 18:27:20 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
Thu Dec 26 00:55:36 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
Wed Dec 25 06:47:36 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
Tue Dec 24 12:31:04 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
Wed Dec 25 06:47:36 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
Mon Dec 23 18:10:16 2019.000000 type 0x0011; code 0x000e; value 0x00000000; Led
....
....
....
```

27.6.2 USB device

27.6.2.1 USB device mass storage

1. 制作盘符

```
# dd if=/dev/zero of=fat32.img bs=1k count=2048
# mkfs.vfat fat32.img
```

2. 将盘符加入

```
# echo fat32.img > /sys/kernel/config/usb_gadget/demo/functions/mass_storage.0/lun.0/file
```

3. 完成后会在 PC 端弹出盘符并支持热插拔

27.6.2.2 USB device adb

1. 启动后无需任何配置直接开始 adb 功能
2. adb 不支持热插拔,插拔后将设备清除,再次插入时需要重新绑定

```
# echo 13500000.otg > /sys/kernel/config/usb_gadget/demo/UDC
```

27.6.2.3 USB device hid

1. pc 端识别设备

```
$ dmesg
```

```
[2180880.531376] usb 2-1.5: new high-speed USB device number 115 using ehci-pci
[2180880.628285] usb 2-1.5: New USB device found, idVendor=18d1, idProduct=d002
[2180880.628292] usb 2-1.5: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[2180880.628304] usb 2-1.5: Product: composite-demo
[2180880.628307] usb 2-1.5: Manufacturer: ingenic
[2180880.628309] usb 2-1.5: SerialNumber: 0123456789ABCDEF
[2180880.635882] input: ingenic composite-demo as
/devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.5/2-1.5:1.0/0003:18D1:D002.006C/input/input126
[2180880.691853] hid-generic 0003:18D1:D002.006C: input,hidraw3: USB HID v1.01 Keyboard [ingenic
composite-demo] on usb-0000:00:1d.0-1.5/input0
```

2. 开发板执行测试程序

```
# cd /testsuite/usb_test/usb_gadget/hid_gadget_test
```

模拟 usb 键盘

```
# ./hid_gadget_test /dev/hidg0 keyboard
```

模拟 usb 鼠标

```
# ./hid_gadget_test /dev/hidg1 mouse
```

27.6.2.4 USB device uvc

1. pc 端识别设备

```
$ dmesg
```

```
[2187560.274157] usb 2-1.5: new high-speed USB device number 19 using ehci-pci
[2187560.371029] usb 2-1.5: New USB device found, idVendor=18d1, idProduct=d002
[2187560.371033] usb 2-1.5: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[2187560.371035] usb 2-1.5: Product: composite-demo
[2187560.371037] usb 2-1.5: Manufacturer: ingenic
[2187560.371039] usb 2-1.5: SerialNumber: 0123456789ABCDEF
[2187560.394857] uvcvideo: Found UVC 1.00 device composite-demo (18d1:d002)
[2187560.403005] input: composite-demo as
/devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.5/2-1.5:1.0/input/input141
```

2. 开发板执行测试程序

webcam_gadget 使用方法：

```
Usage: webcam_gadget [options]
Available options are
-b Use bulk mode
-d Do not use any real V4L2 capture device
-h Print this help screen and exit
-i images dir for [uvc-WxH.jpg uvc-WxH.yuv]
-m Streaming mult for ISOC (b/w 0 and 2)
-n Number of Video buffers (b/w 2 and 32)
-o <IO method> Select UVC IO method:
```

```
                0 = MMAP
                1 = USER_PTR
        -s <speed>      Select USB bus speed (b/w 0 and 2)
                0 = Full Speed (FS)
                1 = High Speed (HS)
                2 = Super Speed (SS)
        -t              Streaming burst (b/w 0 and 15)
        -u device       UVC Video Output device
        -v device       V4L2 Video Capture device
        -e device       HELIX Video Capture device
```

camera 动态测试：

format:

```
# ./webcam_gadget -u /dev/video<uvc video node #> -v /dev/video<camera video node #> -e /dev/video<helix video node #>
```

example:

```
# ./webcam_gadget -u /dev/video7 -v /dev/video4 -e /dev/video0
```

静态图片测试：

format:

```
注：需要提前将测试图片放到-i 指令的路径下，命名规则：uvc-(weight)x(height).yuv 和 uvc-(weight)x(height).jpg
```

example:

```
#./webcam_gadget -u /dev/video<uvc video node #> -v /dev/video<camera video node #> -e /dev/video<helix video node #> -d -i /mnt
```

pc 端测试工具：

```
linux os:
        Video tools: xawtv
        Video tools: cheese webcam booth
windows os:
        Video tools: amcap
手机端:
        APP: usb 摄像头
```

注：华为手机请使用 legacy 配置

27.6.2.5 USB device remote NDIS

1. pc 端识别设备

pc 端识设备信息：

```
$ dmesg
```

识别信息：

```
[2188325.480982] usb 2-1.5: new high-speed USB device number 21 using ehci-pci
```

```
[2188325.577912] usb 2-1.5: New USB device found, idVendor=18d1, idProduct=d002
[2188325.577919] usb 2-1.5: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[2188325.577930] usb 2-1.5: Product: composite-demo
[2188325.577932] usb 2-1.5: Manufacturer: ingenic
[2188325.577934] usb 2-1.5: SerialNumber: 0123456789ABCDEF
[2188325.586180] rndis_host 2-1.5:1.0 usb0: register 'rndis_host' at usb-0000:00:1d.0-1.5, RNDIS device,
52:ea:19:19:5f:69
[2188325.650205] rndis_host 2-1.5:1.0 enp0s29u1u5: renamed from usb0
[2188325.670351] IPv6: ADDRCONF(NETDEV_UP): enp0s29u1u5: link is not ready
```

pc 端识别到 usb 网卡设备：

```
$ ifconfig -a
```

```
enp0s29u1u5 Link encap:Ethernet  HWaddr 52:ea:19:19:5f:69
          UP BROADCAST RUNNING 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)
```

2. pc 端配置 usb 网卡设备 ip：

```
$ sudo ifconfig enp0s29u1u5 192.168.4.250 up
```

```
enp0s29u1u5 Link encap:Ethernet  HWaddr 52:ea:19:19:5f:69
          inet addr:192.168.4.250  Bcast:192.168.4.255  Mask:255.255.255.0
          inet6 addr: fe80::50ea:19ff:fe19:5f69/64 Scope:Link
          UP BROADCAST RUNNING 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)
```

3. pc 端测试网卡

```
$ ping 192.168.4.250
```

```
PING 192.168.4.250 (192.168.4.250) 56(84) bytes of data.
64 bytes from 192.168.4.250: icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from 192.168.4.250: icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from 192.168.4.250: icmp_seq=3 ttl=64 time=0.031 ms
```

27.6.2.6 USB device serial

1. pc 端识别设备

```
$ dmesg
```

设备信息：

```
[2189021.818872] usb 2-1.5: new high-speed USB device number 25 using ehci-pci
[2189021.915645] usb 2-1.5: New USB device found, idVendor=0525, idProduct=a4a7
[2189021.915651] usb 2-1.5: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[2189021.915655] usb 2-1.5: Product: Gadget Serial v2.4
[2189021.915659] usb 2-1.5: Manufacturer: Linux 4.4.94+ with 13500000.otg
[2189021.922892] cdc_acm 2-1.5:2.0: ttyACM0: USB ACM device
```

2. pc 端配置 usb serial 的 VID 和 PID

```
$ echo 0x18d1 0xd002 > /sys/bus/usb-serial/drivers/generic/new_id
```

3. 开发板与 pc 端交互测试

pc 端测试：

```
注：ttyUSB1 对应 usb serial 设备
$ echo 1111111111 > /dev/ttyUSB1
$ cat /dev/ttyUSB1
```

开发板测试：

```
# cat /dev/ttyGS0
# echo 222 > /dev/ttyGS0
```

27.6.2.7 USB device printer

1. pc 端识别设备

```
$ dmesg
```

设备信息：

```
usblp 2-1.7:1.4: usblp0: USB Bidirectional printer dev 38 if 4 alt 0 proto 2 vid 0x18D1 pid 0xD002
```

2. 开发板与 pc 端交互测试

开发板测试：

```
# cd /testsuite/usb_test/usb_gadget/prn_example/
# ./prn_example -read_data
# cat data_file | ./prn_example -write_data
```

pc 端测试：

```
$ echo 111 > /dev/usb/lp0
$ cat /dev/usb/lp0
```

27.6.2.8 USB device uac1.0

1. pc 端识别设备

```
$ aplay -l
```

声卡信息：

```
card 1: compositedemo [composite-demo], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

2. 开发板执行测试程序

```
注：配置 speaker 音频通道，其中声卡 1 为 usb 声卡设备
# amixer cset name='LOO_MUX' LI8
# arecord -f dat -t wav -D hw:1,0 | aplay -D hw:0,0 &
```

3. pc 端选择 usb 声卡设备，播放声音

27.6.3 USB legacy

27.6.3.1 USB device serial

1. pc 端识别设备

```
$ dmesg
```

设备信息：

```
[2189021.818872] usb 2-1.5: new high-speed USB device number 25 using ehci-pci
[2189021.915645] usb 2-1.5: New USB device found, idVendor=0525, idProduct=a4a7
[2189021.915651] usb 2-1.5: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[2189021.915655] usb 2-1.5: Product: Gadget Serial v2.4
[2189021.915659] usb 2-1.5: Manufacturer: Linux 4.4.94+ with 13500000.otg
[2189021.922892] cdc_acm 2-1.5:2.0: ttyACM0: USB ACM device
```

2. 互发信息

pc 端：

```
$echo 1111111111 > /dev/ttyACM0
$cat /dev/ttyACM0
```

开发板端：

```
#cat ttyGS0
#echo 2222 > ttyGS0
```

27.6.3.2 USB device uvc

1. pc 端识别设备

```
$ dmesg
```

```
设备信息：
[1278801.857403] usb 2-1.7: new high-speed USB device number 25 using ehci-pci
[1278801.954312] usb 2-1.7: New USB device found, idVendor=1d6b, idProduct=0102
[1278801.954318] usb 2-1.7: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[1278801.954322] usb 2-1.7: Product: Webcam gadget
[1278801.954326] usb 2-1.7: Manufacturer: Linux Foundation
[1278801.976022] uvcvideo: Found UVC 1.00 device Webcam gadget (1d6b:0102)
[1278801.983318]                input:                Webcam                gadget                as
/devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.7/2-1.7:1.0/input/input119
```

2. 手机端测试方法

a) 手机通过转接线连接到开发板

b) 手机打开 APP：USB 摄像头

注：如果手机识别到设别，请重启开发板

3. 详细测试方法

同上：[device-uvc-gadget]

27.7 注意事项

27.7.1 修改 usb gadget 配置

修改默认支持的 gadget 设备，配置文件路径如下：

```
buildroot/package/ingenic/system_config/usb-device/S90usb
```

当使用某个 function 时，需要将其它功能用’#’注释上：

注：hid 和 serial 只能单独使用，不支持复合设备

```
/etc/init.d/usb/uvc      $1
/etc/init.d/usb/adb      $1
/etc/init.d/usb/mass_storage $1
#/etc/init.d/usb/hid     $1
/etc/init.d/usb/printer $1
/etc/init.d/usb/rndis    $1
/etc/init.d/usb/uac1     $1
#/etc/init.d/usb/serial $1
```

27.7.2 华为手机使用 uvc 时，otg 无法识别问题

使用 legacy 配置，可以解决。

28GMAC 千兆以太网控制器接口

28.1 模块功能介绍

RGMII（Reduced Gigabit Media Independent Interface）是 Reduced GMII（吉比特介质独立接口）。RGMII 均采用 4 位数据接口，工作时钟 125MHz，并且在上升沿和下降沿同时传输数据，因此传输速率可达 1000Mbps。RGMII 数据结构符合 IEEE 以太网标准，接口定义见 IEEE 802.3-2000， RGMII 支持 10/100/1000 兆的总线接口速度。

RMII Reduced Media Independent Interface 简化媒体独立接口，是 IEEE 802.3u 标准中除 MII 接口之外的另一种实现。RMII 支持 10 兆和 100 兆的总线接口速度。

28.2 驱动源码位置

内核驱动代码位置：

```
drivers/net/ethernet/ingenic
   ├── ingenic_mac.c
   ├── synopGMAC_Dev.c
   ├── synopGMAC_plat.c
```

28.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

网卡接口可以配置为 RGMII 和 RMII，在 SDK 中默认配置为 RGMII 接口，用户可以根据自己实际的情况配置为 RMII 接口

双网卡控制器定义：

```
mac0: mac@0x134b0000 {
        compatible = "ingenic,x2000-v12-mac";
        reg = <0x134b0000 0x2000>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_GMAC0>;
        status = "disabled";
        ingenic,rst-ms = <10>;
};
mac1: mac@0x134a0000 {
        compatible = "ingenic,x2000-v12-mac";
        reg = <0x134a0000 0x2000>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_GMAC1>;
        status = "disabled";
        ingenic,rst-ms = <10>;
};
```

28.3.1 设备树默认配置

Halley5_V1.x 系列开发板支持双网卡

Halley5_V2.x 系列开发板支持单网卡 MAC1。

设备树属性说明如下:

属性名称	说明
● ingenic,rst-gpio	定义复位 GPIO
● ingenic,rst-ms	定义复位时间
● ingenic,rst-delay-ms	复位后延时时间，延时期间不对 phy 进行任何操作
● ingenic,mac-mode	RGMII 或者 RMII
● ingenic,mode-reg	
● ingenic,rx-clk-delay	仅 RGMII 模式根据实际情况调整时钟采样偏移
● ingenic,tx-clk-delay	仅 RGMII 模式
● ingenic,phy-clk-freq	如果使用芯片供给 phy 的工作时钟， 需要配置 phy 芯片工作时钟频率

注意:

1.依据开发板设计更改 mac 设备对应 phy 的复位 gpio,复位有效电平,复位需要的保持的时间

2.如果工作在 RGMII 模式下需要配置 TXCLK 和 RXCLK 的 delay,配置的精度是 19.5ps,配置值范围 0-128,delay 的时间范围 0-2.5ns.具体配置值需要由 phy 来确定,TXCLK 和 RXCLK 和 data 保证 2ns 左右的 delay,所以如果 phy 端已经做了 delay,mac 控制器端就不需要设置或补足差值

双网卡板级 RGMII 配置如下

```
&mac0 {
        pinctrl-names = "default", "reset";
        pinctrl-0 = <&mac0_rgmii_p0_normal>, <&mac0_rgmii_p1_normal>, <&mac0_phy_clk>;
        pinctrl-1 = <&mac0_rgmii_p0_rst>, <&mac0_rgmii_p1_normal>, <&mac0_phy_clk>;
        status = "okay";
        ingenic,rst-gpio = <&gpc 22 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
        ingenic,rst-ms = <10>;
        ingenic,rst-delay-ms = <15>;
        ingenic,mac-mode = <RGMII>;
        ingenic,mode-reg = <0xb00000e4>;
        ingenic,rx-clk-delay = <0x2>;
        ingenic,tx-clk-delay = <0x3f>;
        ingenic,phy-clk-freq = <25000000>;
};
```

```
&mac1 {
        pinctrl-names = "default", "reset";
        pinctrl-0 = <&mac1_rgmii_p0_normal>, <&mac1_rgmii_p1_normal>, <&mac1_phy_clk>;
        pinctrl-1 = <&mac1_rgmii_p0_rst>, <&mac1_rgmii_p1_normal>, <&mac1_phy_clk>;
```

```
        status = "okay";
        ingenic,rst-gpio = <&gpb 16 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
        ingenic,rst-ms = <10>;
        ingenic,rst-delay-ms = <15>;
        ingenic,mac-mode = <RGMII>;
        ingenic,mode-reg = <0xb00000e8>;
        ingenic,rx-clk-delay = <0x0>;
        ingenic,tx-clk-delay = <0x3f>;
        ingenic,phy-clk-freq = <25000000>;
};
```

28.3.2 设备树自定义配置

RMII 配置参考如下:

```
&mac0 {
                pinctrl-names = "default", "reset";
                pinctrl-0 = <&mac0_rmii_p0_normal>, <&mac0_rmii_p1_normal>, <&mac0_phy_clk>;
                pinctrl-1 = <&mac0_rmii_p0_rst>, <&mac0_rmii_p1_normal>;
                status = "okay";
                ingenic,rst-gpio = <&gpb 0 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,rst-ms = <10>;
                ingenic,rst-delay-ms = <15>;
                ingenic,mac-mode = <RMII>;
                ingenic,mode-reg = <0xb00000e4>;
                ingenic,phy-clk-freq = <25000000>;
};
```

```
&mac1 {
                pinctrl-names = "default", "reset";
                pinctrl-0 = <&mac1_rmii_p0_normal>, <&mac1_rmii_p1_normal>, <&mac1_phy_clk>;
                pinctrl-1 = <&mac1_rmii_p0_rst>, <&mac1_rmii_p1_normal>;
                status = "disable";
                ingenic,rst-gpio = <&gpb 26 GPIO_ACTIVE_LOW INGENIC_GPIO_NOBIAS>;
                ingenic,rst-ms = <10>;
                ingenic,rst-delay-ms = <15>;
                ingenic,mac-mode = <RMII>;
                ingenic,mode-reg = <0xb00000e8>;
                ingenic,phy-clk-freq = <25000000>;
};
```

28.4 内核编译配置

内核配置 INGENIC_MAC，配置说明如下：

```
Symbol: INGENIC_MAC [=y]
Type  : tristate
```

```
rompt: ingenic on-chip MAC support
  Location:
    -> Device Drivers
      -> Network device support (NETDEVICES [=y])
        -> Ethernet driver support (ETHERNET [=y])
  Defined at drivers/net/ethernet/ingenic/Kconfig:1
  Depends on: NETDEVICES [=y] && ETHERNET [=y]
  Selects: CRC32 [=y] && MII [=y]
```

28.4.1 内核默认编译配置

内核默认配置 GMAC 驱动，配置界面如下：

```
--- Ethernet driver support
<*>   ingenic on-chip MAC support
[*]     Ingenic mac dma interfaces
          Ingenic mac dma bus interfaces (MAC_AXI_BUS)  --->
```

28.4.2 内核自定义编译配置

可配置选项	配置说明
Ingenic gmac receive descriptor number[80..10240]	gmac 控制器接收数据包的缓存数量，缓存越大可以减少满载时的丢包率，经过测试设置 8192 可以避免控制器层出现丢包， 这种情况运行时一个网卡的动态内存会使用 16M-32M
Dual core mutex transmission	使用双网卡时配置， 配置后会把双网卡处接收数据协议栈的过程互斥处理，可以减少指令 cache 的 miss 率

28.5 设备节点生成

驱动加载成功后，执行 ifconfig -a 会出现 eth0 或者 eth1.

28.6 应用程序使用说明

执行 ifconfig -a 命令查看支持的网络设备，如果出现 eth0 或者 eth1 则说明网卡加载成功

```
# ifconfig -a
eth0          Link encap:Ethernet  HWaddr 1E:01:7F:E6:D3:26
              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)

eth1          Link encap:Ethernet  HWaddr 3E:4C:9D:F4:74:4B
```

```
                  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)

lo              Link  encap:Local  Loopback
                  LOOPBACK    MTU:65536    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:1
                  RX  bytes:0  (0.0  B)    TX  bytes:0  (0.0  B)
```

配置网络 IP

```
ifconfig  eth0  IP                              /*根据实际选择的网口配置 eth0 或 eth1*/
```

使用 ping 命令查看网络连接情况

```
ping  IP  -I  eth0                              /*通过-I  选项可以指定 ping  命令使用的网口*/
```

28.6.1 网络性能测试

找一台有千兆以太网功能的电脑，使用网线直连，使用 iperf3 命令测试，电脑当做服务端，开发板做客户端

- 电脑端执行命令

```
iperf3    -s
```

- 测试单向发送性能

```
iperf3    -c  192.168.4.105  -u  -b  1000M  -l  65507  -t  10
```

- 测试单向接收性能

```
iperf3    -c  192.168.4.105  -u  -b  1000M  -l  65507  -t  10  -R
```

- 应用层提高网络性能的方法

  以下方法根据具体应用场景使用，设置不合理反而会影响测试结果

```
sysctl  -w  net.core.rmem_default=10485760
```

■ 增加网络核心层接收的缓存， 此设置会使用 10M 内存， 双网口桥接不使用

```
sysctl  -w  net.core.rmem_max=10485760
```

■ 增加网络核心层接收的缓存

指定内核处理接收网络协议栈使用的 cpu 核，指定后可以避免内核动态切换过程造成双核使用不均衡造成的性能不稳定：

```
echo  0  >  /proc/irq/61/smp_affinity_list
```

- 指定 cpu0 处理 gmac1 接收的数据

```
echo  1  >  /proc/irq/63/smp_affinity_list
```

- 指定 cpu1 处理 gmac0 接收的数据， gmac0 中断号 63 gmac1 中断号 61

```
sysctl  -w  net.core.netdev_max_backlog=4000
```

- 增加转接过程中数据包缓存长度，可以减少数据抖动但消耗内存越大

```
echo  2  >  /sys/class/net/eth0/queues/rx-0/rps_cpus
```

- 指定 cpu1 处理 eth0 接收到数据包，输入 1 指定 cpu0，2 指定 cpu1， 3 指定 cpu0-1

28.6.2 1588 硬件时间戳测试

使用 2 块带有以太网接口的开发板，使用网线直连。

使用 CONFIG_INGENIC_GMAC_USE_HWSTAMP 宏配置 kernel 的 1588 功能。

文件系统中需要有测试 1588 功能的测试程序，默认使用 linuxptp 测试，linuxptp 测试工程在 manhatton 工程的 buildroot 中，关于 linuxptp 使用方法可以参照网络资料。

网络正常工作后：
一个开发板做 slave 执行命令：

```
ptp4l  -E  -4  -H  -i  eth0  -s  -m
```

另一个开发板做 master 执行命令：

```
ptp4l  -E  -4  -H  -i  eth0  -m
```

29AES 加解密驱动接口

29.1 模块功能介绍

- aes 对称加解密硬件模块
- Rijdael_fips_197 标准
- 支持 128/192/256 CBC/ECB 加解密

29.2 驱动源码位置

内核驱动所在路径：

```
$ drivers/crypto/ingenic-aes.c
```

29.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

AES 控制器定义：

```
aes: aes@0x13430000 {
        compatible = "ingenic,aes";
        reg = <0x13430000 0x10000>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_AES>;
        status = "ok";
};
```

29.3.1 设备树默认配置

设备树默认编译会产生 AES 设备。

29.3.2 设备树自定义配置

用户可根据实际需求关闭 AES 设备，在板级.dts 中将该节点配置为 disabled。

```
&aes  {
      status  =  "disabled";
  };
```

29.4 内核编译配置

编译选项：CRYPTO_DEV_INGENIC_AES，配置说明如下：

```
Symbol:  CRYPTO_USER_API_SKCIPHER  [=y]
Type    :  tristate
Prompt:  User-space  interface  for  symmetric  key  cipher  algorithms
   Location:
          ->  Cryptographic  API  (CRYPTO  [=y])
     Defined  at  crypto/Kconfig:1616
```

```
    Depends on: CRYPTO [=y] && NET [=y]
    Selects: CRYPTO_BLKCIPHER [=y] && CRYPTO_USER_API [=y]
```

```
Symbol: CRYPTO_DEV_INGENIC_AES [=y]
Type  : tristate
Prompt: Support for INGENIC AES hw engine
  Location:
    -> Cryptographic API (CRYPTO [=y])
      -> Hardware crypto devices (CRYPTO_HW [=y])
  Defined at drivers/crypto/Kconfig:311
  Depends on: CRYPTO [=y] && CRYPTO_HW [=y] && (MARCH_XBURST1 || MACH_XBURST2 [=y])
  Selects: CRYPTO_AES [=y] && CRYPTO_BLKCIPHER2 [=y]
```

29.4.1 内核默认编译配置

内核默认配置 AES 驱动。

29.4.2 内核自定义编译配置

用户可根据实际需求配置 AES 驱动，配置界面如下：

```
--- Hardware crypto devices
<*>   Support for INGENIC AES hw engine
< >   Imagination Technologies hardware hash accelerator
<*>   Support for Ingenic SHA hw accelerator
```

29.5 设备节点生成

驱动加载成功后生成以下节点

```
/sys/bus/platform/drivers/aes
```

29.6 应用程序使用说明

测试应用程序路径

```
packages/example/Sample/security_utils/aes/
├── aes.c
├── CMakeLists.txt
├── gen_key_base.c
├── include
│   ├── aes.h
│   ├── bignum.h
│   ├── gen_key_base.h
│   └── keys.h
└── main.c
```

测试方法：

```
aes_test  in_file  out_file key_file  op(en:1  de:0)
```

相关测试数据：

```
packages/example/Sample/security_utils/test_data/aes_test/
├── aes_test_data.txt
├── aes_test_en.txt
└── key.txt
```

- key.txt:key 数据
- aes_result.txt：该数据为结果数据
- aes_test_data.txt：该数据为待测试数据

使用示例：

```
#  hash_test  aes_test_data.txt  aes_result.txt key.txt  1
```

30RSA 加解密驱动接口

30.1 模块功能介绍

RSA 公开密钥密码体制是一种使用不同的加密密钥与解密密钥，支持 1024/2048 密钥长度加解密

30.2 驱动位置

驱动源码所在位置

```
$ drivers/misc/ingenic_rsa.c
```

30.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

RSA 控制器定义：

```
rsa: rsa@0x13480000 {
            compatible = "ingenic,rsa";
            reg = <0x13480000 0x10000>;
            interrupt-parent = <&core_intc>;
            interrupts = <IRQ_RSA>;
            status = "ok";
 };
```

30.3.1 设备树默认配置

设备树默认编译会产生 rsa 设备

30.3.2 设备树自定义配置

用户可根据实际需求关闭 rsa 设备，在板级.dts 中将该节点配置为 disabled。

```
&rsa {
      status = "disabled";
 };
```

30.4 内核编译配置

内核配置选项 INGENIC_RSA，配置说明如下：

```
Symbol: INGENIC_RSA [=y]
Type  : boolean
Prompt: JZ RSA Driver
      Location:
            -> Device Drivers
      Defined at drivers/misc/Kconfig:536
      Depends on: MACH_XBURST2 [=y]
```

30.4.1 内核默认编译配置

内核默认未配置 RSA 驱动。

30.4.2 内核自定义编译配置

用户可根据实际需求可配置 RSA 驱动，配置界面如下：

```
< > Analog Devices Digital Potentiometers
< > Dummy IRQ handler
< > Integrated Circuits ICS932S401
< > Enclosure Services
< > Medfield Avago APDS9802 ALS Sensor module
< > Intersil ISL29003 ambient light sensor
< > Intersil ISL29020 ambient light sensor
< > Taos TSL2550 ambient light sensor
< > ROHM BH1780GLI ambient light sensor
< > BH1770GLC / SFH7770 combined ALS - Proximity sensor
< > APDS990X combined als and proximity sensors
< > Honeywell HMC6352 compass
< > Dallas DS1682 Total Elapsed Time Recorder with Alarm
< > BMP085 digital pressure sensor on I2C
< > FSA9480 USB Switch
<*> Bluetooth power control driver for BCM-4345C5 module
[ ] Generic on-chip SRAM driver
[*] JZ RSA Driver
[ ] Linux pmem allocator
< > Silicon Labs C2 port support  ----
    EEPROM support  --->
    Texas Instruments shared transport line discipline  --->
< > STMicroeletronics LIS3LV02Dx three-axis digital accelerometer (I2C)
    *** Altera FPGA firmware download module ***
< > Altera FPGA firmware download module
    *** Intel MIC Bus Driver ***
    *** SCIF Bus Driver ***
    *** Intel MIC Host Driver ***
    *** Intel MIC Card Driver ***
    *** SCIF Driver ***
    *** Intel MIC Coprocessor State Management (COSM) Drivers ***
< > Line Echo Canceller support
```

30.5 设备节点生成

驱动加载成功后生成以下节点：

```
/dev/rsa
```

30.6 应用程序使用说明

测试程序路径

```
packages/example/Sample/security_utils/rsa
├── CMakeLists.txt
├── gen_key_base.c
├── include
│   ├── bignum.h
│   ├── gen_key_base.h
│   ├── jz_rsa.h
```

```
|       |——— keys.h
|       └——— rsa.h
|——— main.c
|——— make.sh
|——— rsa.c
|——— sec_test.c
```

测试方法:

```
rsa_test  in_file out_file rsa_key
```

其中：in_file 为需要运算的数据

out_file 为运算完成生成的结果数据

rsa_key 为 key 数据文件

相关测试数据文件位置:

```
packages/example/Sample/security_utils/test_data/rsa_test/
|——— rsa_key
|——— rsa_result.txt
└——— rsa_test_data.txt
```

- rsa_key:为 rsa 密钥
- rsa_result.txt：该数据为结果数据
- rsa_test_data.txt：该数据为待测试数据

执行测试

```
rsa_test  rsa_test_data.txt  rsa_result.txt  rsa_key
```

31Hash 模块驱动接口

31.1 模块功能介绍

支持 hash 算法加速，支持 SHA1 RFC-3174 标准和 MD5 RFC-1321 标准。

支持 SHA 160/224/256/384/512

支持 MD5 128bit

31.2 驱动源码位置

驱动源码所在位置：

```
drivers/crypto/ingenic-hash.c
```

31.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

Hash 控制器定义

```
hash: hash@0x13470000 {
        compatible = "ingenic,hash";
        reg = <0x13470000 0x10000>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_HASH>;
        status = "ok";
};
```

31.3.1 设备树默认配置

设备树默认编译会产生 hash 设备。

31.3.2 设备树自定义配置

用户可根据实际需求管不 hash 设备，将该节点配置为 disabled。

31.4 内核编译配置

内核驱动编译选项 CRYPTO_DEV_INGENIC_SHA

```
Symbol: CRYPTO_USER_API_HASH [=y]

Type    : tristate

Prompt: User-space interface for hash algorithms
```

```
  Location:
        -> Cryptographic API (CRYPTO [=y])

  Defined at crypto/Kconfig:1607

  Depends on: CRYPTO [=y] && NET [=y]

  Selects: CRYPTO_HASH [=y] && CRYPTO_USER_API [=y]

Symbol: CRYPTO_DEV_INGENIC_SHA [=y]

Type    : tristate

Prompt: Support for Ingenic SHA hw accelerator
  Location:
    -> Cryptographic API (CRYPTO [=y])

        -> Hardware crypto devices (CRYPTO_HW [=y])
  Defined at drivers/crypto/Kconfig:493
  Depends on: CRYPTO [=y] && CRYPTO_HW [=y] && (MARCH_XBURST1 || MACH_XBURST2 [=y])
  Selects: CRYPTO_ALGAPI [=y]
```

31.4.1 内核默认编译配置

内核默认编译会产生 Hash 控制器设备。配置界面如下：

```
--- Hardware crypto devices
< >   Support for INGENIC AES hw engine
< >   Imagination Technologies hardware hash accelerator
<*>   Support for Ingenic SHA hw accelerator
```

31.4.2 内核自定义编译配置

用户可根据实际需求关闭该设备。

31.5 设备节点生成

31.6 应用程序使用说明

参考应用测试程序路径：

```
packages/example/Sample/security_utils/hash/
├── CMakeLists.txt
├── hash.c
├── hash_openssl.c
├── include
│   ├── bignum.h
```

```
/      ├── hash.h
/      └── hash_openssl.h
└── main.c
```

测试方法：

```
hash_test  in_file  out_file
```

相关测试数据：

```
packages/example/Sample/security_utils/test_data/hash_test/
├── hash_result.txt
└── hash_test_data.txt
```

- hash_result.txt：该数据为结果数据
- hash_test_data.txt：该数据为待测试数据

```
#  hash_test  rsa_test_data.txt  hash_result.txt
```

32PWM 模块驱动接口

32.1 模块功能介绍

Pulse width modulation (pwm)模块设计分为 16 个独立的通道，可以互不影响的进行独立工作，同时又分为两种工作模式，为 cpu 模式和 DMA 模式。cpu 模式通过使用输入的时钟源来产生方波，而 DMA 模式的方波产生是通过 fifo 中的的数据来决定周期以及占空比，另外 DMA 模式必须按照 4word 进行对齐。

32.2 驱动源码位置

驱动源码所在位置：

```
drivers/pwm/pwm-ingenic-v2.c
```

32.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12-pinctrl.dtsi
```

1.定义相关 PWM IO Function

```
pwm_pin:  pwm-pin{
                        pwm0_pc:  pwm0_pc  {
                                ingenic,pinmux  =  <&gpc  0  0>;
                                ingenic,pinmux-funcsel  =  <PINCTL_FUNCTION0>;
                        };
....
....
....
                       pwm15_pc:  pwm15_pc  {
                                ingenic,pinmux  =  <&gpc  15  15>;
                                ingenic,pinmux-funcsel  =  <PINCTL_FUNCTION0>;
                        };
```

2.定义 PWM 控制器功能

```
pwm:  pwm@0x134c0000  {
            compatible  =  "ingenic,x2000-pwm";
  #pwm-cells = <2>;
            reg  =  <0x134c0000  0x10000>;
            interrupt-parent  =  <&core_intc>;
            interrupts  =  <IRQ_PWM>;
};
```

32.3.1 设备树默认配置

内核 halley5_v20.dts 板级配置文件中默认使用了 pwm0_pd 作为 PWM 输出功能

```
&pwm {
        pinctrl-names = "default";
        pinctrl-0 = <&pwm0_pd>;
        status = "okay";
};
```

32.3.2 设备树自定义配置

如果需要使用其他通道的 GPIO 作为 PWM，可以修改 halley5_v20.dts 文件,比如添加 pwm1_pc

```
&pwm {
        pinctrl-names = "default";
        pinctrl-0 = <&pwm0_pd>, <&pwm1_pc>;
        status = "okay";
};
```

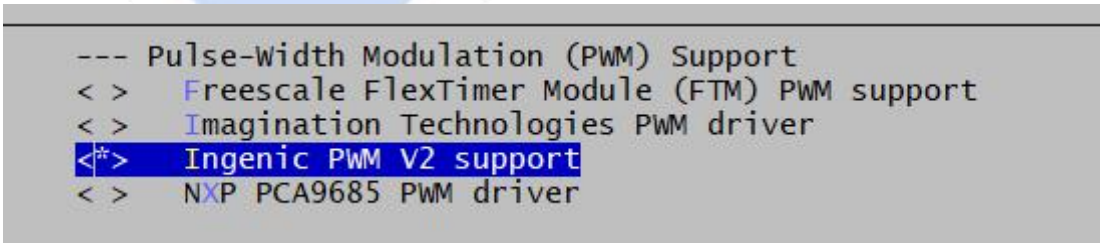
32.4 内核编译配置

内核驱动编译选项使用 PWM_INGENIC_V2 控制，配置说明如下：

```
Symbol: PWM_INGENIC_V2 [=y]
Type  : tristate
Prompt: Ingenic PWM V2 support
  Location:
    -> Device Drivers
      -> Pulse-Width Modulation (PWM) Support (PWM [=y])
  Defined at drivers/pwm/Kconfig:202
  Depends on: PWM [=y] && MACH_XBURST2 [=y]
```

32.4.1 内核默认编译配置

内核默认配置 PWM 驱动，配置界面如下：



32.4.2 核自定义编译配置

如果产品不需要 PWM 功能，可以去掉该选项以减小内核大小。

32.5 设备节点生成

可以通过标准/sys/class/pwm 查看 pwm 设备的具体情况，通过标准 sysfs 接口，进行 pwm 用户空间的导出和使用。

32.6 应用程序使用说明

1. 进入/sys/class/pwm/pwmchip0 目录下：

```
# ls
device     export     npwm       power      subsystem  uevent     unexport
```

2. 查看 npwm 可获取当前支持的 pwm 通道数，目前总共支持 16 个：

```
# cat npwm
16
```

3. 通过写 0～15 到 export 可导出对应的 pwm 进行使用，下面以 pwm0 为例：

```
# echo 0 > export
```

然后可以看到当前目录下多出了一个 pwm0 目录，其他通道类似：

```
# ls
device     npwm       pwm0       uevent
export     power      subsystem  unexport
```

4. 进入到 pwm0 目录，目前可以设置两项，一个是占空比 duty_cycle，另外一个是周期 period：

```
# ls
duty_cycle  enable      period      polarity    power       uevent
```

5. 比如，设置周期为 1000000000，单位为纳秒：

```
# echo 1000000000 > period
[58044.902196] period=23809523
[58044.905080] duty=0 level_init=1 low=23809523
```

6. 设置占空比为 500000000，单位为纳秒：

```
# echo 500000000 > duty_cycle
[58210.268663] period=23809523
[58210.271585] duty=11904761 level_init=1 low=11904762
```

7. 使能 pwm0：

```
# echo 1 > enable
```

如果设置正确，此时会有对应的波形输出。

8. 如果不再使用可以关闭波形输出：

```
# echo 0 > enable
```

9. 最后也可以将 pwm0 导出关闭：

```
# cd ..
# ls
device     npwm       pwm0       uevent
export     power      subsystem  unexport
# echo 0 > unexport
# ls
device     export     npwm       power      subsystem  uevent     unexport
```

可以看到此时 pwm0 目录已经没有了。

33DTRNG 模块驱动接口

33.1 模块功能介绍

真随机数发生器，用于在加解密场景提供真随机数的功能

33.2 驱动源码位置

驱动源码所在位置：

```
drivers/char/hw_random/ingenic-rng.c
```

33.3 设备树配置

设备树所在位置：

```
arch/mips/boot/dts/ingenic/x2000-v12.dtsi
```

设备树描述：

```
dtrng: dtrng@0x10072000 {
        compatible = "ingenic,dtrng";
        reg = <0x10072000 0x100>;
        interrupt-parent = <&core_intc>;
        interrupts = <IRQ_DTRNG>;
        status = "disabled";
};
```

33.3.1 设备树默认配置

默认编译会产生 dtrng 设备。

```
&dtrng {
        status = "okay";
};
```

33.3.2 设备树自定义配置

如果客户需要关闭 dtrng 设备，可以将以上节点配置为 disabled。

```
&dtrng {
        status = "disabled";
};
```

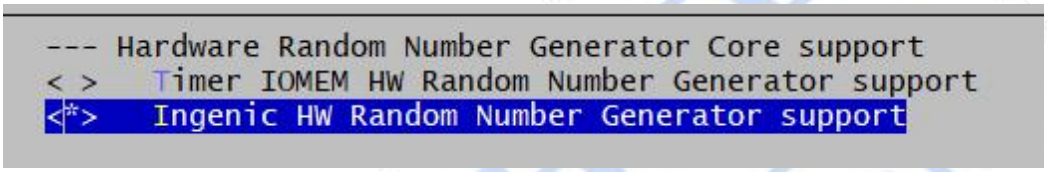
33.4 内核编译配置

内核配置 HW_RANDOM_INGENIC，配置说明如下：

```
Symbol: HW_RANDOM_INGENIC [=y]
Type  : tristate
Prompt: Ingenic HW Random Number Generator support
  Location:
    -> Device Drivers
      -> Character devices
(1)     -> Hardware Random Number Generator Core support (HW_RANDOM [=y])
  Defined at drivers/char/hw_random/Kconfig:384
  Depends on: HW_RANDOM [=y] && MIPS [=y] && (SOC_X2000_V12 [=y] || SOC_M300 [=n])
```

33.4.1 内核默认编译配置

内核默认配置 dtrng 驱动，配置界面如下：



33.4.2 内核自定义编译配置

客户可以根据实际情况，去掉该驱动的配置。

33.5 设备节点生成

当驱动加载成功后，会生成/dev/hwrng，用户可以根据此节点判断驱动是否成功加载。

33.6 应用程序使用说明

可以通过以下应用程序，测试随机数。

```
#include<stdio.h>
#include<stdlib.h>
#include<linux/watchdog.h>
#include<sys/ioctl.h>
#include<sys/types.h>
#include<fcntl.h>
#include<unistd.h>
#include<errno.h>

int  main(int  argc,  char  *argv[])
{
        int  fd  ,retval;
        char  buf[64]  =  {0};
        int  cmd  =  0;
        if  (!argv[1])
        {
                printf("argv[1]  is  NULL  \n");
```

```
                exit(errno);
        }
        fd = open(argv[1],O_RDONLY);
        if(fd < 0)
        {
                printf("open %s false\n",argv[1]);
                exit(errno);
        }
        if(read(fd,buf,32) < 0)
                printf(" read false\n");
        else
                printf("random_num register value %d\n",*(unsigned int *)buf);

        close(fd);
        return retval;
}
```

测试结果

```
# ./hwrng /dev/hwrng
random_num register value 9363246
# ./hwrng /dev/hwrng
random_num register value 1329736129
# ./hwrng /dev/hwrng
random_num register value 567871915
# ./hwrng /dev/hwrng
random_num register value 966768232
```