Table of Contents

前言

文档目的及背景

君正处理器是高集成度、高性能和低功耗的 32 位 RISC 处理器，带有 MMU 和数据及指令Cache，以及丰富的外围设备，可以运行 Linux 操作系统。本文将向读者介绍基于君正处理器平台进行 Linux 内核的配置方法和开发过程，,引导开发人员快速进行 Linux 开发。本文档为君正内核 3.10 版本开发文档，基于芯片 X1830，不针对具体开发板，文中如有涉及具体开发板型号，是为了说明方便。

在阅读该文档前，需要具备以下基本技能：
1.会使用 Linux 系统进行开发，最好是 ubuntu。
2.知道嵌入式开发基本流程。如 uboot，linux，文件系统制作等。

阅读该文档，会提供以下帮助：
1.帮助理解君正 BSP 基本组成。（uboot,linux,文件系统）
2.提供基于君正开发平台创建自己的应用程序方法。
3.提供应用程序访问驱动的基本测试用例。

Linux内核驱动和应用

本章主要介绍内核的各个模块的驱动和相应的用户空间测试方法。通过阅读本章节，旨在对引导内核编译，对内核的各个驱动有基本的了解，对相应的测试程序有一定的了解。

以下介绍的模块，在发布的内核源码中不一定全部包含，可以根据需要，按照文档的说明，自己添加相关驱动的编译配置。

本章节不限定在某个固定的平台，但是为了描述方便，会以 halley3 nand开发板为例进行介绍。

内核配置和编译

在君正发布的BSP中，会根据发布的开发板型号组成 defconfig。内核的 defconfig 一般组成格式如下：

```
[board_name]_[media_type]_linux_defconfig
[board_name]:  发布开发板名。
[media_type]: 一般是开发板所使用的存储介质名。例如，nor,spinand 等。
```

具体使用哪一个作为开发板的默认配置，按照发布为准。

在 arch/mips/configs/目录下可以找到相应的配置文件。

例如，针对 halley3_nand_v11 的开发板，内核提供的默认配置为

```
halley3_v11_sfcnand_ubi_defconfig
```

在 PC 开发环境下执行

```
$ make halley3_v11_sfcnand_ubi_defconfig

$ make xImage
```

会生成 arch/mips/boot/compressed/xImage

可以在 defconfig 的基础上，通过 make menuconfig 添加自己的驱动模块，或为内核添加其它的特性。例如：

```
$ make halley3_v11_sfcnand_ubi_defconfig

$ make menuconfig
```

Xburst 板级介绍

在发布的内核版本中，针对不同的芯片型号，会在arch/mips/xburst目录下进行添加，该目录基本介绍如下：

```
common/ #所有芯片公共部分
core/ # xburst 核心文件
Kconfig
lib/
Makefile
Platform
soc-m200/  #m200 系列板级
soc-x1000/ #x1000 系列板级
soc-x1830/ #其板级信息定义在ingenic/路径下
```

以halley3开发板为例，其板级定义在ingenic/x1830/boards/halley3

该目录重要文件介绍如下：

```
├── common
│   ├── 43438_bt_power_control.c        #蓝牙电源管理
│   ├── 43438_wlan_device.c
│   ├── 43438_wlan_power_control.c      #wifi电源管理
│   ├── i2c_bus.c                       #内核i2c设备描述
│   ├── lcd                             #不同LCD屏幕参数配置
│   │   ├── lcd-hy035jt.c
│   │   ├── lcd-kd035c10hc010a.c
│   │   ├── lcd-vs035hst31c1.c
│   │   └── Makefile
│   ├── Makefile
│   ├── pwm_generic.c                   #pwm的定义
│   ├── rtc.c
├── halley3_v11
│   ├── board.h                         #所有使用的 GPIO 定义。
│   ├── Makefile
│   ├── pm.c
├── Kconfig
└── Makefile
```

其他相关信息定义在ingenic/x1830/boards/common

```
├── board_base.c                        #各个platform_device注册
├── board_base.h
├── Makefile
├── misc.c
```

```
├── mmc.c                                   #sd控制器设备描述
├── sensor_board.c                          #camera 定义
├── sfc_bus.c
├── sfc_nor_table.h
├── speaker.c
├── spi_bus.c                               #内核spi设备描述
```

GPIO模块

x1830的gpio控制器有五组，A, B, C, D, Z。

支持输入输出和设备复用功能。 内核的gpio驱动程序是基于gpio子系统架构编写的。应用程序可以使用gpio_demo进行测试。内核驱动可以在内核空间使用，也可以通过导出gpio sys节点到用户空间，在用户空间进行操作。

文件介绍

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/xburst/soc-x1830/include/mach/platform.h

在arch/mips/xburst/soc-x1830/common/platform.c 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子系统实现的， 所以其他驱动程序可以通过内核提供的libgpio接口，很方便的进行gpio控制，例如 gpio_request_one, gpio_get_value, gpio_set_value等。

gpio 驱动文件所在位置：

```
arch/mips/xburst/soc-x1830/common/gpio.c
```

编译配置

-*- GPIO Support /*内核通过配置 CONFIG_GPIOLIB 选项可以使用 gpio 功能，默认必须选上*/

```
--- GPIO Support

[]   Debug GPIO calls

/*如果要将 gpio 导出到用户节点/sys/class/gpio 下，对该节点下的文件操作，可以配置以下选项*/

[*]   /sys/class/gpio/... \(sysfs interface\)

*** Memory mapped GPIO drivers: ***

< > Generic memory-mapped GPIO controller support (MMIO platform device)
```

< > Dallas's 1-wire support --->

用户空间

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。

```
  /sys/class/gpio/

        "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。

        “unexpect” 导出到用户空间的逆操作。

                 例如：“echo 19; unexpect” 将会移除使用“export”文件导出的“gpio19”节点。
```

GPIO信号的路径似 /sys/class/gpio/gpio42/ (对于GPIO #42 来说)，并有如下的读写属性：

/sys/class/gpio/gpioN/

"direction" 读取得到“in” 或 “out”。这个值通常运行写入。

“edge”读取得到“none”、“rising”、“failing”或者“both”。

“value” 读取得到0（低电平）或1（高电平)。

“active_low” 读取得到 0 （假）或 1 （真）。

SPI模块

x1830 的 spi 控制器为 SSI，SSI 支持的接口有 Microwire，SSP，SPI。SPI 接口可连接至 spi nor，支持 spi 读写功能。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。Spi 总线下，可以挂普通的char设备，也可以挂 mtd 设备。可以在内核空间通过 spi驱动操作 spi 接口，也可以在用户空间通过 spi_ioc_transfer 操作 spi 接口。`

文件介绍

```
内核驱动基于 spi 驱动架构,所有的硬件资源在板级定义。

按照 spi 驱动编写规范，需要提供 spi_board_info。

以下介绍驱动对应在内核中的路径
    板级资源定义路径:
        ingenic/x1830/board/common/spi_bus.c，这个文件下面是 spi 的设备。

与 X1830 spi 相关的驱动所在目录和文件说明,忽略目录中存在的其他文件。
drivers/spi/
    ├── jz_spi.c
    ├── jz_spi.h
    ├── spi-bitbang.c
    ├── spi.c
    ├── spidev.c
```

编译配置

```
    一般发布的软件版本中会默认配置 spi 驱动,如果需要自己更改 spi 的编译选项,可以通过以下方式进行配置。
在工作电脑上执行:
    $ make menuconfig

配置以下选项:
--- SPI support
  [ ]   Debug support for SPI drivers
          *** SPI Master Controller Drivers ***
  < >   Altera SPI Controller
  < >   Ingeinc JZ47XX SPI controller test driver
  <*>   Ingenic JZ series SPI driver
  [ ]     Ingenic SoC SSI controller 0 for SPI Host driver
  [*]     Ingenic SoC SSI controller 1 for SPI Host driver
  [ ]       Disable DMA (always use PIO) on JZ SSI controller 1
              JZ SSI1 controller function pins select (PORT C 19BIT)  --->
  [*]     Board info associated by spi master
  [ ]     Use GPIO CE on JZ SSI controller 0
```

```
  < >     Ingenic spi test driver
  [ ]     Ingenic SoC SSI controller select external clk
  -*-   Utilities for Bitbanging SPI masters
          *** SPI Protocol Masters ***
      /*配置以下选项，可以把 spi 的设备节点导出到/dev 下，供用户空间使用*/
  <*>   User mode SPI device driver support
  < >   Infineon TLE62X0 (for power switching)
```

测试应用

在Documentation/spi/spidev_test.c中， 通过更改device编号（eg：/dev/spidev1.1），然后将编译生成的二进制文件导入文件系统并执行测试。

```
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>

#define ARRAY_SIZE(a) (sizeof(a) / sizeof((a)[0]))

static void pabort(const char *s)
{
        perror(s);
        abort();
}

static const char *device = "/dev/spidev1.1";
static uint8_t mode;
static uint8_t bits = 8;
static uint32_t speed = 500000;
static uint16_t delay;

static void transfer(int fd)
{
        int ret;
        uint8_t tx[] = {
                0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
                0x40, 0x00, 0x00, 0x00, 0x00, 0x95,
                0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
                0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
                0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
                0xDE, 0xAD, 0xBE, 0xEF, 0xBA, 0xAD,
                0xF0, 0x0D,
```

```
        };
        uint8_t rx[ARRAY_SIZE(tx)] = {0, };
        struct spi_ioc_transfer tr = {
                .tx_buf = (unsigned long)tx,
                .rx_buf = (unsigned long)rx,
                .len = ARRAY_SIZE(tx),
                .delay_usecs = delay,
                .speed_hz = speed,
                .bits_per_word = bits,
        };

        ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
        if (ret < 1)
                pabort("can't send spi message");

        for (ret = 0; ret < ARRAY_SIZE(tx); ret++) {
                if (!(ret % 6))
                        puts("");
                printf("%.2X ", rx[ret]);
        }
        puts("");
}

static void print_usage(const char *prog)
{
        printf("Usage: %s [-DsbdlHOLC3]\n", prog);
        puts("  -D --device   device to use (default /dev/spidev1.1)\n"
             "  -s --speed    max speed (Hz)\n"
             "  -d --delay    delay (usec)\n"
             "  -b --bpw      bits per word \n"
             "  -l --loop     loopback\n"
             "  -H --cpha     clock phase\n"
             "  -O --cpol     clock polarity\n"
             "  -L --lsb      least significant bit first\n"
             "  -C --cs-high  chip select active high\n"
             "  -3 --3wire    SI/SO signals shared\n");
        exit(1);
}

static void parse_opts(int argc, char *argv[])
{
        while (1) {
                static const struct option lopts[] = {
                        { "device",  1, 0, 'D' },
                        { "speed",   1, 0, 's' },
                        { "delay",   1, 0, 'd' },
                        { "bpw",     1, 0, 'b' },
                        { "loop",    0, 0, 'l' },
                        { "cpha",    0, 0, 'H' },
                        { "cpol",    0, 0, 'O' },
                        { "lsb",     0, 0, 'L' },
```

```
                        { "cs-high", 0, 0, 'C' },
                        { "3wire",   0, 0, '3' },
                        { "no-cs",   0, 0, 'N' },
                        { "ready",   0, 0, 'R' },
                        { NULL, 0, 0, 0 },
                };
                int c;

                c = getopt_long(argc, argv, "D:s:d:b:lHOLC3NR", lopts, NULL);

                if (c == -1)
                        break;

                switch (c) {
                case 'D':
                        device = optarg;
                        break;
                case 's':
                        speed = atoi(optarg);
                        break;
                case 'd':
                        delay = atoi(optarg);
                        break;
                case 'b':
                        bits = atoi(optarg);
                        break;
                case 'l':
                        mode |= SPI_LOOP;
                        break;
                case 'H':
                        mode |= SPI_CPHA;
                        break;
                case 'O':
                        mode |= SPI_CPOL;
                        break;
                case 'L':
                        mode |= SPI_LSB_FIRST;
                        break;
                case 'C':
                        mode |= SPI_CS_HIGH;
                        break;
                case '3':
                        mode |= SPI_3WIRE;
                        break;
                case 'N':
                        mode |= SPI_NO_CS;
                        break;
                case 'R':
                        mode |= SPI_READY;
                        break;
                default:
```

```
                        print_usage(argv[0]);
                        break;
                }
        }
}

int main(int argc, char *argv[])
{
        int ret = 0;
        int fd;

        parse_opts(argc, argv);

        fd = open(device, O_RDWR);
        if (fd < 0)
                pabort("can't open device");

        /*
         * spi mode
         */
        ret = ioctl(fd, SPI_IOC_WR_MODE, &mode);
        if (ret == -1)
                pabort("can't set spi mode");

        ret = ioctl(fd, SPI_IOC_RD_MODE, &mode);
        if (ret == -1)
                pabort("can't get spi mode");

        /*
         * bits per word
         */
        ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
        if (ret == -1)
                pabort("can't set bits per word");

        ret = ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits);
        if (ret == -1)
                pabort("can't get bits per word");

        /*
         * max speed hz
         */
        ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
        if (ret == -1)
                pabort("can't set max speed hz");

        ret = ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed);
        if (ret == -1)
                pabort("can't get max speed hz");

        printf("spi mode: %d\n", mode);
```

```
        printf("bits per word: %d\n", bits);
        printf("max speed: %d Hz (%d KHz)\n", speed, speed/1000);

        transfer(fd);

        close(fd);

        return ret;
}
```

蓝牙-WIFI模块

WIFI、蓝牙使用的是AP6214A芯片，WIFI使用SDIO接口进行通信，而蓝牙使用uart接口进行通信。

编译配置(buildroot)

一般发布的软件版本中都会默认配置WIFI和Bluetooth测试应用， 如需要自己更改Bluetooth和WiFi的编译选项，可以通过以下方式进行配置。buildroot中的开发板配置文件在configs下。

1. 配置过程

根据开发板选用相应的配置文件，halley3选用halley3_v10_defconfig配置。

```
$ make halley3_v10_defconfig

$ make menuconfig
```

2. 配置选项

2.1 蓝牙（Bluetooth）配置

```
--- ingenic packages
 [*]   bluetooth
         --- bluetooth
            select bluebooth chip  --->
             [*] BCM43438
                   select BCM43438 firmware (BCM43438_A1)  --->
                   select bluetooth protocol (bluez)  --->
             [ ] RTK_8723
         (/dev/ttyS1) bluetooth device
         [ ]   pulseaudio config for bluetooth
         [ ]   play to bluetooth
         (halley3) prefix of bluetooth device name
 [*]   bluetooth manager
 [ ]   bluetooth media control demo
 [*]   bluez auto agent for demo
 -*-   bluez alsa
```

2.2 WIFI配置

```
[*]   wifi
```

```
        --- wifi
        [ ]   common wifi device
                  --select wifi chip
                          [*] WIFI_BCM
                                      WIFI BCM Default Select (WIFI_BCM_43438)  ---
>
                          [ ] WIFI_RTK
        -*-   wifi use wpa_supplicant
        (/etc/wpa_supplicant.conf) wpa_supplicant.conf file path
        (/usr/data/hostapd.conf) hostapd.conf file path
        (wlan0) hostap interface
        [*]     wifi auto startup
        [*]     wifi use ap mode
        (C8:93:46:45:C3:80) wifi mac address
        (/data/misc/wifi/) wifi mac address path
```

3.Target packages配置

```
Location:
   -> Target packages
     -> Networking applications
-*- bluez-utils 5.x
[ ] build OBEX support
-*- build CLI client
-*- install deprecated tool
-*- dnsmasq
[*] dhcp support
-*- hostapd
-*-   Enable WPS
[*] ntp
[*]   ntpdate
[*] tcpdump
[*]   smb dump support
-*- wpa_supplicant
[*]   Enable nl80211 support
-*-   Install wpa_cli binary
-*-   Install wpa_client shared library
-*-   Install wpa_passphrase binary
```

命令使用

蓝牙

>蓝牙匹配流程

1>使用hciconfig -a查看蓝牙信息

```
# hciconfig -a
hci0:   Type: Primary  Bus: UART
        BD Address: D0:31:10:AF:EB:02  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING PSCAN ISCAN
        RX bytes:1231 acl:0 sco:0 events:63 errors:0
        TX bytes:1360 acl:0 sco:0 commands:63 errors:0
        Features: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87
        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
        Link policy: RSWITCH SNIFF
        Link mode: SLAVE ACCEPT
        Name: 'halley3_EB02'
        Class: 0x040414
        Service Classes: Rendering
        Device Class: Audio/Video, Loudspeaker
        HCI Version: 4.1 (0x7)  Revision: 0xe7
        LMP Version: 4.1 (0x7)  Subversion: 0x2209
        Manufacturer: Broadcom Corporation (15)
```

2>连接设备开启蓝牙，搜索 halley3_EB02，点击该设备进行配对

```
AutoAgent: [NEW] Device 9C:A5:C0:20:26:A7 vivo X6S A
GattServer: Device connected
AutoAgent: Accept pairing (yes/no): yes
udhcpc: sending discover
[BT] property changed: Modalias
[BT] property changed: UUIDs
[BT] property changed: ServicesResolved
[BT] property changed: Paired
[BT] INFO: device connected
====================>call dev_connect_callback 1
call dev_connect_callback fail ,m_isAvailable is false
AutoAgent: [CHG] Device 9C:A5:C0:20:26:A7 vivo X6S A Modalias: bluetooth:v001Dp1200
d1436
AutoAgent: [CHG] Device 9C:A5:C0:20:26:A7 vivo X6S A UUIDs: 00001103-0000-1000-8000
-00805f9b34fb
AutoAgent: [CHG] Device 9C:A5:C0:20:26:A7 vivo X6S A ServicesResolved: yes
AutoAgent: [CHG] Device 9C:A5:C0:20:26:A7 vivo X6S A Paired: yes
AutoAgent: Authorize service 0000110d-0000-1000-8000-00805f9b34fb (yes/no): yes
[BT] property changed: UUIDs
AutoAgent: [CHG] Device 9C:A5:C0:20:26:A7 vivo X6S A UUIDs: 00001103-0000-1000-8000
-00805f9b34fb
AutoAgent: [CHG] Device 9C:A5:C0:20:26:A7 vivo X6S A UUIDs: 00001105-0000-1000-8000
-00805f9b34fb
AutoAgent: [CHG] Device 9C:A5:C0:20:26:A7 vivo X6S A UUIDs: 00001106-0000-1000-8000
-00805f9b34fb
[BT] connected bluetooth MediaControl1 /org/bluez/hci0/dev_9C_A5_C0_20_26_A7/fd0
bluealsa: bluez.c:767: Endpoint method call: org.bluez.MediaEndpoint1.SetConfigurat
ion()
bluealsa: ctl.c:667: Sending notification: 1 => 14
```

```
bluealsa-aplay: aplay.c:735: Fetching available transports
bluealsa: bluez.c:696: A2DP Sink (SBC) configured for device 9C:A5:C0:20:26:A7
bluealsa: bluez.c:698: Configuration: channels: 2, sampling: 44100
bluealsa: transport.c:746: State transition: 0 -> 0
bluealsa-aplay: aplay.c:800: Creating PCM worker 9C:A5:C0:20:26:A7
bluealsa: bluez.c:867: Registering endpoint: /A2DP/SBC/Sink/2
bluetoothd[181]: Endpoint registered: sender=:1.2 path=/A2DP/SBC/Sink/2
bluealsa-aplay: ../src/shared/ctl-client.c:105: Connecting to socket: /var/run/blue
alsa/hci0
bluealsa: ctl.c:636: Received new connection: 16
bluealsa: ctl.c:649: New client accepted: 16
bluealsa-aplay: ../src/shared/ctl-client.c:375: Requesting PCM open for 9C:A5:C0:20
:26:A7
bluealsa: ctl.c:343: PCM requested for 9C:A5:C0:20:26:A7 type 1 stream 1
bluealsa-aplay: aplay.c:385: Starting PCM loop
udhcpc: sending discover
bluetoothd[181]: Can't open input device: No such file or directory (2)
bluetoothd[181]: AVRCP: failed to init uinput for 9C:A5:C0:20:26:A7
[BT] connected bluetooth player /org/bluez/hci0/dev_9C_A5_C0_20_26_A7/player0
bluealsa: bluez.c:1314: Signal: PropertiesChanged: org.bluez.MediaPlayer1: Repeat
bluealsa: bluez.c:1314: Signal: PropertiesChanged: org.bluez.MediaPlayer1: Shuffle
bluealsa: bluez.c:1314: Signal: PropertiesChanged: org.bluez.MediaPlayer1: Status
bluealsa: bluez.c:1314: Signal: PropertiesChanged: org.bluez.MediaPlayer1: Track
bluealsa: bluez.c:1314: Signal: PropertiesChanged: org.bluez.MediaPlayer1: Position
```

>常用命令

- bt_up.sh启动蓝牙（默认启动执行）
- bt_down.sh关闭蓝牙
- hciconfig -a查看蓝牙信息
- bluetoothctl命令

1>执行bluetoothctl命令

```
# bluetoothctl
[NEW] Controller D0:31:10:AF:EB:02 halley3_EB02 [default]
[NEW] Device 9C:A5:C0:20:26:A7 vivo X6S A
Agent registered
[vivo X6S A]#
```

2>输入help显示以下属性(列举部分)

```
[vivo X6S A]# help
Available commands:
  list                       List available controllers
  show [ctrl]                Controller information
  select <ctrl>              Select default controller
  devices                    List available devices
  paired-devices             List paired devices
```

```
  system-alias <name>        Set controller alias
  reset-alias                Reset controller alias

  scan <on/off>              Scan for devices
  info [dev]                 Device information
  pair [dev]                 Pair with device
  trust [dev]                Trust device
  untrust [dev]              Untrust device
  block [dev]                Block device
  unblock [dev]              Unblock device
  remove <dev>               Remove device
  connect <dev>              Connect device
  disconnect [dev]           Disconnect device
  list-attributes [dev]      List attributes
  set-alias <alias>          Set device alias
  select-attribute <attribute/UUID>
                             Select attribute
  attribute-info [attribute/UUID]
                             Select attribute
  read                       Read attribute value
  write <data=[xx xx ...]>   Write attribute value
  acquire-write              Acquire Write file descriptor
  release-write              Release Write file descriptor
  acquire-notify             Acquire Notify file descriptor
  release-notify             Release Notify file descriptor
  notify <on/off>            Notify attribute value

  version                    Display version
  quit                       Quit program
  exit                       Quit program
  help                       Display help about this program
```

WiFi

>WiFi匹配流程

1>启动后，配置/etc/wpa_supplicant.conf中WiFi名字（ssid）和密码（psk）

```
# cat etc/wpa_supplicant.conf
ctrl_interface=/var/run/wpa_supplicant
update_config=1
country=GB"

network={
    ssid="test"
    scan_ssid=1
    psk="123456ab"
    bssid=
    priority=1
```

```
}
```

2>配置完成后执行wifi_up.sh

```
# wifi_up.sh
[ 3109.423516] BCM:
[ 3109.423516] Dongle Host Driver, version 1.141.66
[ 3109.423516] Compiled in drivers/net/wireless/bcmdhd on Aug 15 2019 at 10:18:32
[ 3109.437878] BCM:wl_android_wifi_on in
[ 3109.441679] BCM:wifi_platform_set_power = 1
[ 3109.446035] BCM:========== WLAN placed in RESET ON  ========
[ 3109.468504] sdio_reset_comm():
[ 3109.510720] mmc1: queuing unknown CIS tuple 0x80 (2 bytes)
[ 3109.519490] mmc1: queuing unknown CIS tuple 0x80 (3 bytes)
[ 3109.528253] mmc1: queuing unknown CIS tuple 0x80 (3 bytes)
[ 3109.539463] mmc1: queuing unknown CIS tuple 0x80 (7 bytes)
[ 3109.551901] mmc1: queuing unknown CIS tuple 0x81 (9 bytes)
[ 3109.726455] BCM:F1 signature OK, socitype:0x1 chip:0xa9a6 rev:0x1 pkg:0x4
[ 3109.734014] BCM:DHD: dongle ram size is set to 524288(orig 524288) at 0x0
[ 3109.741272] concate_revision: chip:a9a6
[ 3109.745264] fw_path:/lib/firmware/wifi_bcm_43438/fw_43438_a1 nv_path:/lib/firmwa
re/wifi_bcm_43438/nv_43438_a1
[ 3109.798742] BCM:dhdsdio_write_vars: Download, Upload and compare of NVRAM succee
ded.
[ 3109.860463] BCM:dhd_bus_init: enable 0x06, ready 0x06 (waited 0us)
[ 3109.867795] BCM:wifi_platform_get_mac_addr
[ 3109.873532] BCM:dhd_get_concurrent_capabilites: Get P2P failed (error=-23)
[ 3109.880663] BCM:Firmware up: op_mode=0x0001, MAC=28:ed:e0:94:8b:b1
[ 3109.891291] BCM:dhd_preinit_ioctls pspretend_threshold for HostAPD failed  -23
[ 3109.903149] BCM:Firmware version = wl0: Jun 27 2017 11:07:16 version 7.46.57.4.i
g.r5d4 (A1 Station/P2P) FWID 01-bd86c2ef es7.c5.n4.a3.ap1
[ 3109.916651] BCM:dhd_wlfc_hostreorder_init(): successful bdcv2 tlv signaling, 64
# Successfully initialized wpa_supplicant
udhcpc: started, v1.25.1
udhcpc: sending discover
nl80211: Could not re-add multicast membership for vendor events: -2 (No such file
or directory)
udhcpc: sending discover
[ 3129.713940] BCM:6CFG80211-ERROR) wl_cfg80211_connect : BCM:Connectting withff:ff
:ff:ff:ff:ff channel (0) ssid "test", len (4)
[ 3130.829290] BCM:wl_bss_connect_done succeeded with 9c:a5:c0:20:26:a8
[ 3130.840666] BCM:wl_bss_connect_done succeeded with 9c:a5:c0:20:26:a8
udhcpc: sending discover
udhcpc: sending select for 192.168.43.150
udhcpc: lease of 192.168.43.150 obtained, lease time 3600
deleting routers
adding dns 192.168.43.1
```

3>使用ping www.baidu.com验证网络连接成功

```
# ping www.baidu.com
PING www.baidu.com (61.135.169.121): 56 data bytes
64 bytes from 61.135.169.121: seq=0 ttl=55 time=33.770 ms
64 bytes from 61.135.169.121: seq=1 ttl=55 time=50.481 ms
64 bytes from 61.135.169.121: seq=2 ttl=55 time=50.138 ms
64 bytes from 61.135.169.121: seq=3 ttl=55 time=50.936 ms
64 bytes from 61.135.169.121: seq=4 ttl=55 time=53.542 ms
```

4>使用wifi_down.sh关闭WiFi

Camera驱动配置

x1830 camera摄像头控制器为isp，x1830平台使用的接口为dvp和mipi，内核的sensor驱动是基于isp驱动框架编写的，应用程序可以使用isp_demo进行测试。

内核编译配置

```
--- Multimedia support
        *** Multimedia core support ***
  [*]   Cameras/video grabbers support
  [ ]   Analog TV support
  [ ]   Digital TV support
  [ ]   Remote Controller support
  [*]   Media Controller API
  [*]   V4L2 sub-device userspace API
  [ ]   Enable advanced debug functionality on V4L2 drivers
  [ ]   Enable old-style fixed minor ranges on drivers/video device
  <*>   V4L2 int device (DEPRECATED)
        *** Media drivers ***
  [*]   Media USB Adapters           /*USB camera*/
        --- Media USB Adapters
              *** Webcam devices ***
         <*>   USB Video Class (UVC)
         [*]     UVC input events device support
  [*]   V4L platform devices
         --- V4L platform devices
         [*]   TX ISP MODULE
         [*]     TX ISP LIBISP
         [*]     MCLK from CPU
  [ ]   Memory-to-memory multimedia devices  --->
  [ ]   Media test drivers  --->
         *** Supported MMC/SDIO adapters ***
  < >   Cypress firmware helper routines
        *** Media ancillary drivers (tuners, sensors, i2c, frontends) ***
  [ ]   Autoselect ancillary drivers (tuners, sensors, i2c, frontends)
        Encoders, decoders, sensors and other helper chips  --->
        Sensors used on soc_camera driver  --->
        Sensors used on tx-isp_camera driver  --->
            *** tx-isp camera sensor drivers ***
            < > gc0328 camera support
            < > sc2235 camera support
            <*> OV9712 camera support
            < > gc2375a camera support
            < > ov9732 camera support
        Customise DVB Frontends  --->
```

编译配置（Buildroot）

```
BR2_PACKAGE_ISP

isp demo

Symbol: BR2_PACKAGE_ISP_DEMO [=y]
Type  : boolean
Prompt: isp demo
   Location:

      -> ingenic packages (BR2_PACKAGE_INGENIC [=y])
   Defined at package/ingenic/isp_demo/Config.in:1
   Depends on: BR2_PACKAGE_INGENIC [=y]
   Selects: BR2_PACKAGE_LIBISP [=y]

BR2_PACKAGE_LIBISP:

libisp
  Symbol: BR2_PACKAGE_LIBISP [=y]
  Type  : boolean
  Prompt: libisp
    Location:
       -> ingenic packages (BR2_PACKAGE_INGENIC [=y])
    Defined at package/ingenic/libisp/Config.in:1
    Depends on: BR2_PACKAGE_INGENIC [=y]
    Selected by: BR2_PACKAGE_ISP_DEMO [=y] && BR2_PACKAGE_INGENIC [=y]
```

测试应用

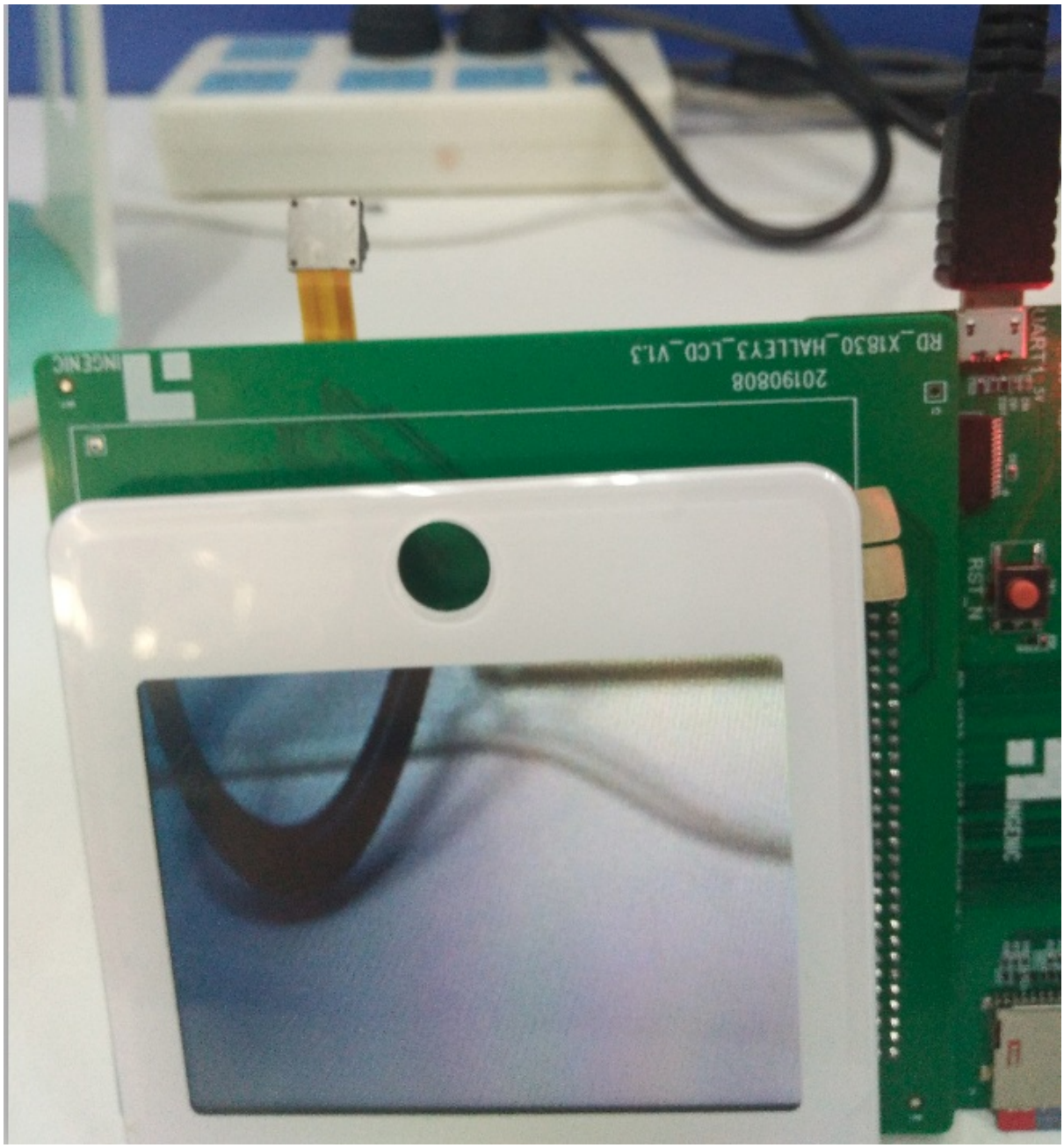
在x1830的camera测试应用是isp_demo(路径：/usr/bin/isp_demo),针对不同的sensor需要手动添加对应的宏到isp_demo.c

```
#include <imp_inc/isp/isp_tuning.h>
#include <imp_inc/fb/fb_manager.h>

#include "image_convert.h"


#define LOG_TAG                         "isp_demo"

//#define SENSOR_OV9712
#define SENSOR_GC2375A


#if defined SENSOR_SC2235
#define SENSOR_NAME                     "sc2235"
#define SENSOR_CUBS_TYPE                TX_SENSOR_CONTROL_INTERFACE_I2C
#define OUTPUT_PIX_FMT                  PIX_FMT_NV12
#define SENSOR_I2C_ADDR                 0x30
#define SENSOR_WIDTH                    640
#define SENSOR_HEIGHT                   1072
#elif defined SENSOR_OV9712
```

开发板端执行 isp_demo:

```
# isp_demo
[ 3090.412447] set sensor gpio as PA-low-10bit
[ 3090.423071] ov9712 chip found @ 0x30 (i2c0)
1970-01-01 08:51:30.424856:I/IMP - isp_cim 430: SENSOR INPUT: [0] ov9712
1970-01-01 08:51:30.458456:I/IMP - isp_cim 298: Image width: 1280 height: 800 size:
 1536000
1970-01-01 08:51:30.458844:I/IMP - rmem 100: CMD Line Rmem Size:34603008, Addr:0x05
f00000
```

LCD屏显示sensor的拍摄的实时画面



Audio模块

alsa 库

```
通过配置buildroot的menuconfig文件， 选择对应的alsa库，
```

> alsa 工具使用

```
amixer, aplay, arecord 的使用方法如下：
假设播放的音频文件为 play.wav，录音生成的文件为 record.wav，在开发板上按照以下操作。
(1) 播放音频文件
    $ aplay play.wav

(2) 通过 amic 录音
    $ arecord -D hw:0,0 -c 2 -f S16_LE -r 44100 -d 10 record.wav

    单独使用该命令可以在当前目录下生成 10 秒的录音文件 record.wav。一般在执行 arecord
命令之前，会通过 amixer 设置录音的通道和参数，命令如下：
    $ amixer cset numid=1,iface=MIXER,name='Master Playback Volume' 15
    $ amixer cset numid=2,iface=MIXER,name='Mic Volume' 3

(3) 通过 dmic 录音
    通过 dmic 录 8K 采样率四通道音频数据：
    $ arecord -D hw:0,1 -c 4 -f S16_LE -r 8000 -d 10 record.wav

(4)audio 测试工具说明：
    1.arecord 录音
        -D 参数用于指定音频设备PCM
        hw 的第一个参数用来指定声卡号，第二个参数用于指定设备号
        -c 用于指定声道数
        -f 用于指定数据格式
        -r 用于指定采样频率
        -d 用于指定录音时间
        --help 获取帮助
    2.aplay 放音
        参数设置与 arecord 一致。
```

Watchdog应用

命令说明

```
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
```

命令运行

```
watchdog -t 30 /dev/watchdog
```

执行命令后30ms重启系统。