君正®

IMP T31 BSP 开发指南 V1.0

Date: 2022-04

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

君正®
IMP T31 BSP 开发指南 V1.0

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.

北京君正集成电路股份有限公司

地址: 北京市海淀区西北旺东路 10 号院东区 14 号楼君正大厦
电话:(86-10)56345000
传真:(86-10)56345001
Http: //www.ingenic.cn

前言

概述

本文为 Ingenic-T31 开发指南使用说明，旨在帮助用户了解 T31 开发板的功能及开发流程 。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
T31	

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	版本	修订章节
2019-07	1.0	第一次正式版本发布
2019-10	1.1	第二次发布
2021-08	1.2	第三次发布
2021-09	1.3	第四次发布
2022-04	2.0	**第 6 章** 6.12 WiFi 小节涉及新增

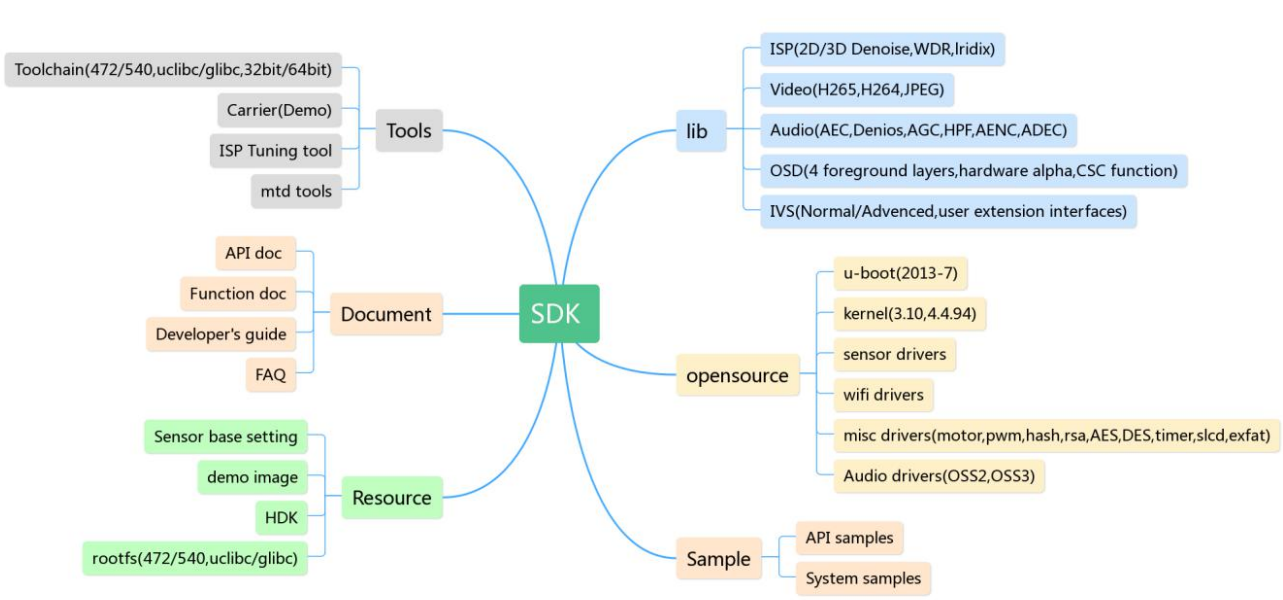
目录

1 开发资源介绍

1.1 SDK 介绍

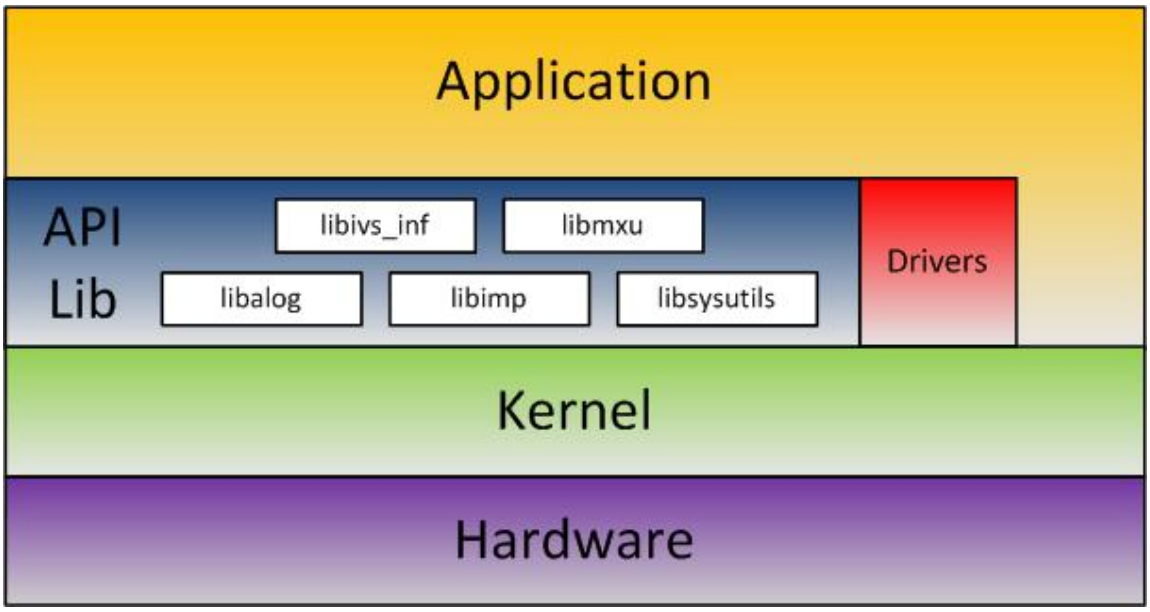
ISVP SDK，即软件开发工具包，包括 API 库、开源源码、文档、Samples 等。开发者可以通过 SDK 快速的开展产品功能开发。以下是 ISVP SDK 的内容概览图：

图 1-1 SDK 组成结构



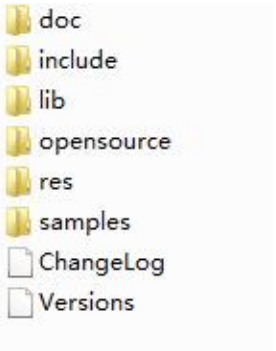
1.2 SDK 层次结构

图 1-2 SDK 层次机构



- Hardware: 硬件层，完成 I/O 等具体的硬件功能。
- Linux Kernel: 内核层，完成基础的系统功能，定义硬件资源。
- drivers: ko 模块驱动，可通过 driver 进行硬件操作。
- API lib: 接口库，实现硬件功能的抽象，方便于应用层的开发。API 库主要有五部分：
  - libimp: 多媒体功能库。如 H264 编码，JPEG 编码，IVS 和 Audio 等。
  - libsysutils: 系统功能库。如重启，设置系统时间和电池功能等。
  - libalog: ISVP-SDK 的 log 实现库。
  - libivs_inf: IVS 算法库，包括越线检测，周界防范等。
  - libmxu: 128 位 mxu 加速指令算子库。
- Application: 应用层。实现功能逻辑等。
  - Application 推荐使用 SDK 库提供的 API 及配合 drivers 进行开发。对于一些特殊的功能需求，也可以直接调用内核接口进行开发。

1.3 SDK LIB



doc：指导文档
include：相关头文件
lib：相关库文件
opensource：kernel、uboot、drivers、busybox 等
res：文件镜像、效果文件等
samples：相关应用代码

2 开发环境搭建

2.1 为什么需要搭建开发环境

由于嵌入式单板的资源有限，不能在单板上运行开发和调试工具，通常需要交叉编译调试的方式进行开发和调试，即“宿主机＋目标机”的形式。宿主机和目标机一般采用串口连接显示交互信息，网口连接传输文件。

但宿主机和目标机的处理器一般不相同。宿主机需要建立适合于目标机的交叉编译环境。程序在宿主机上经过“编译－连接－定位”得到可执行文件。通过一定的方法将可执行文件烧写到目标机中，然后在目标机上运行。

2.2 Toolchain 安装使用

2.2.1 Toolchain 是什么

Toolchain 即交叉编译工具链，是 Linux Host 机上用来编译和调试嵌入式设备程序的一系列工具的集合。ISVP 中的 Toolchain 版本信息如下：

1.Toolchain472:

1) gcc 版本：4.7.2
2) libc 版本：
   A. glibc 版本：2.16
   B. uclibc 版本：0.9.33.2-nptl

2.Toolchain540:

1) gcc 版本：5.4.0
2) libc 版本：
   A. glibc 版本：2.22
   B. uclibc 版本：0.9.33.2

2.2.2 如何安装 Toolchain

安装流程：

第一步：安装 7z 解压工具`$ sudo apt-get install p7zip`

第二步：根据 Host 机 CPU 位宽选择 mips-gcc472-glibc216-32bit.7z 或者 mips-gcc472-glibc216-64bit.7z 进行解压。例如：

```
$ 7z x mips-gcc472-glibc216-64bit.7z
```

第三步：通过 export PATH=xxxx:$PATH 命令，将 toolchain 下的 bin 目录添加到 PATH 环境变量中或者在~/.bashrc 中加上下面一句永久改变。

```
$ vim ~/.bashrc
$ export PATH=/opt/mips-gcc472-glibc216-64bit/bin:$PATH
```

第四步：测试 toolchain 可执行：

```
$ mips-linux-gnu-gcc --version
mips-linux-gnu-gcc (Ingenic 2015.02) 4.7.2                Copyright (C) 2012Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

若出现如上信息则可确认 toolchain 安装正确。

2.2.3 交叉工具包

交叉工具链是一个软件安装包，是由很多工具的集合。有常用的 gcc 编译工具、objdump 反汇编工具、as 汇编器、ld 链接器、size 查看二进制文件大小等。

3 T31 必读

3.1 Uboot 编译

- Uboot 编译流程：

u-boot 可单独编译，不依赖其他代码。T31 u-boot 的板机配置文件位于 include/configs/isvp_t31.h。默认编译配置文件介绍如表 3-1。

表 3-1 T31 芯片对应 uboot 编译文件

芯片型号	编译命令	说明
T31N	make isvp_t31_sfcnor	表示编译 norflash 启动的 uboot，针对 T31N 芯片
T31L	make isvp_t31_sfcnor_lite	表示编译 norflash 启动的 uboot，针对 T31L 芯片
T31X	make isvp_t31_sfcnor_ddr128M	表示编译 norflash 启动的 uboot，针对 T31X 芯片
T31A	make isvp_t31a_sfcnor_ddr128M	表示编译 norflash 启动的 uboot，针对 T31A 芯片
T31AL	make isvp_t31al_sfcnor_ddr128M	表示编译 norflash 启动的 uboot，针对 T31AL 芯片
T31N	make isvp_t31_msc0	表示编译 SD 卡启动的 uboot，针针对 T31N 芯片
T31L	make isvp_t31_msc0_lite	表示编译 SD 卡启动的 uboot，针针对 T31L 芯片
T31X	make isvp_t31_msc0_ddr128M	表示编译 SD 卡启动的 uboot，针针对 T31X 芯片
T31A	make isvp_t31a_msc0_ddr128M	表示编译 SD 卡启动的 uboot，针针对 T31A 芯片
T31AL	make isvp_t31al_msc0_ddr128M	表示编译 SD 卡启动的 uboot，针对 T31AL 芯片

编译步骤：

第一步： `$ make distclean` 清除旧配置

第二步： `$ make isvp_t31_xxx` 根据相应芯片，编译生成对应的 u-boot-with-spl.bin

Uboot 配置文件中常见修改的地方：

1) CONFIG_BOOTARGS，主要修改点是内核启动以后的内存配置，分区大小配置。（注：mem 表示内核启动以后保留内存，rmem 表示预留给 SDK 的内存（包括 ISP 模块的内存）；两者相加为芯片真实内存大小；具体大小可参考代码）。

2) CONFIG_BOOTCOMMAND，配置 uboot 启动执行的命令。例如：norflash 启动模式下添加 sd 卡启动的命令，`"sf probe;sf read 0x80600000 0x40000 0x280000; bootm 0x80600000"` 改为 `"mmc read 0x80600000 0x1800 0x3000; bootm 0x80600000"`。

3) CONFIG_BOOTDELAY，配置 uboot 的等待时间。

4) 需要添加新的 norflash 芯片的支持。

5) uboot 中添加密码功能：

修改配置文件，修改 isvp_t31.h 中添加如下内容：

`#define CONFIG_AUTOBOOT_KEYED` // 必配

`#define CONFIG_AUTOBOOT_STOP_STR “123456”` //必配，uboot 设置的密码。

`#define CONFIG_AUTOBOOT_PROMPT “Press xxx in %d second”`,bootdelay //选配，uboot 提示信息。

`#define CONFIG_AUTOBOOT_DELAY_STR “linux”` //选配，

uboot 提示信息代码的具体实现在 common/main.c 中 `abortboot_keyed(int bootdelay)`；可以根据自己的需要具体改动。

6) SD 卡升级问题

在 isvp_t31.h 中添加 `#define CONFIG_AUTO_UPDATE` 定义。具体代码实现在 common/cmd_sdupdate.c 中。需要注意：`LOAD_ADDR` 表示把 SD 卡上的相应内存加载到内存的位置。程序中默认设置为 0x82000000，由于这个地址位于 uboot 的堆上，常见 uboot 的堆大小设置在 isvp_t31.h 中 `CONFIG_SYS_MALLOC_LEN` 宏的配置；所以这个地址能够被使用的大小将受到堆大小的限制，还有 uboot 代码中 malloc 空间的限制。（注意：需要读取较大文件的时候，可以适当增大 `CONFIG_SYS_MALLOC_LEN`）

7) 编译出的 uboot 大于限制的大小

uboot 代码中默认限制 spl 部分为 26Kbytes，uboot 部分为 214Kbytes；一共 240Kbytes 大小的限制。如果 uboot 编译生成的 u-boot-with-spl.bin 文件大于 240Kbytes，则无法启动。

解决方案一：

增加 uboot 的限制；修改 isvp_t31.h 中 `CONFIG_SYS_MONITOR_LEN` 宏的定义；同时修改 `CONFIG_BOOTARGS` 变量中 boot 分区的大小及以后分区的偏移地址。

解决方案二：

如果生成的 u-boot-with-spl.bin 超出 240Kbytes 较少可以采取压缩 uboot 的方式。在 isvp_t31.h 中 修改`#undef CONFIG_SPL_LZOP` 为 `#define CONFIG_SPL_LZOP`；然后重新编译烧录的文件名为 u-boot-lzo-with-spl.bin

8) uboot 网络问题

默认的 isvp 配置是包含以太网部分的代码，如果产品在 uboot 阶段不需要 tftpboot 下载或者 NFS 挂载，可以把以太网部分代码裁剪掉，以便减小 uboot。

具体操作：打开 isvp_T31.h 配置文件，把`#define CONFIG_CMD_NET` 宏注掉即可。

3.2 Kernel 编译

kernel 可单独编译，不依赖其他代码。以 isvp 板级编译为例，进入 kernel 源码目录。在 arch/mips/configs/ 文件夹下存放了内核的板机配置文件；T31 芯片板级文件为 isvp_swan_defconfig；

第一步：使用相对配置好的板级文件 `$ make isvp_swan_defconfig`

第二步：根据需求选择性编译 `$ make menuconfig`

第三步：编译内核文件 `$ make uImage`

注意：对于 kernel 之外的 ko 编译，依赖 kernel，且 kernel 必须经过编译。每当 kernel 更改之后，可能会出现 ko 无法 insmod 的情况，或者可以 insmod 但会出现未知错误。这是因为 kernel 重新编译后，函数 Symbol 表有变化，需要重新编译 ko driver 以正确 link 函数。此时需要重新编译 ko。

内核默认的 config 留有一定余量，而实际产品往往需要进行内核裁减以释放更多的空间。

以下是几个常用可供裁减的选项及说明：

1) CONFIG_NETWORK_FILESYSTEMS：网络文件系统，一般用来方便开发，但会占用较多空间，也可用 tftpboot 进行替代。若不支持 NFS，可以 Disable。

2) CONFIG_KALLSYMS &CONFIG_KALLSYMS_ALL：内核函数符号表，会占用较多空间，在 panic 时的函数栈可以显示出函数名。当内核稳定后，可以考虑 Disable 此功能，但建议完全稳定之前使能此功能。

3) 其他文件系统。一般文件系统会占用较多空间，开发者可根据需求对文件系统的选项进行裁减，比如 ext 文件系统等。

4) 和产品定义无关的模块可以裁掉，比如 USB，以太网，TF 卡等。

注：内核的深度裁减有一定的技术难度，开发者尽量在深入了解配置的情况下再进行深度裁减。

3.3 Busybox 编译

与 kernel 的编译方法类似，需要先 make defconfig 再 make，之后 make install 会默认把安装文件生成在 busybox/_install 目录下。

```
make isvp_glibc_defconfig
make ; make install
```

配置说明：

```
isvp_glibc_defconfig  --glibc 默认 config
isvp_glibc_mini_defconfig  --glibc 默认裁剪 config
isvp_uclibc_defconfig  --uclibc 默认 config
isvp_uclibc_mini_defconfig  --uclibc 默认裁剪 config
```

3.4 应用程序编译

应用程序编译注意有以下几点：

1) 关于 glibc 和 uclibc 编译的区分方法请参考《Toolchain 使用说明》
2) 关于 API 库的链接顺序：[IVS 库] [mxu 库] [libimp/libsysutils] [libalog]
3) 由于 libimp 中依赖 C++库，因此需要使用 mips-linux-gnu-g++进行链接，若使用 gcc 链接，需要手动添加 `LD_FLAGS+=stdc++`
4) 如何优化 elf 文件的大小：
   A. 编译等级选择 O2：`C_FLAGS/CXX_FLAGS += -O2`
   B. `DFLAG += -Wl,-gc-sections`，不链接不必要的段。
   C. elf 文件执行 mips-linux-gnu-strip，链接后删除不必要的段。
5) 若系统中有多个 elf 文件需要链接库文件，可使用动态链接的方式，若只有一个文件链接库文件，请使用静态链接的方式（注，libimp 相关功能在系统中只能存在一份实例）。在调试时选择动态链接的方式可以方便的进行 debug 及问题反馈。

3.5 T31 需要加载的驱动

驱动名称	驱动介绍
tx-isp-t31.ko	ISP 驱动
sensor_xxxx_t31.ko	Sensor 驱动
avpu.ko	视频编码驱动
sinfo.ko	Sinfo 探测驱动
audio.ko	音频驱动

4 文件系统

4.1 根文件系统介绍

Linux 的目录结构的最顶层是一个被称为“.”的根目录。系统加载 Linux 内核之后，就会挂载一个设备到根目录上。存在于这个设备中的文件系统被称为根文件系统。所有的系统命令、系统配置以及其他文件系统的挂载点都位于这个根文件系统中。

根文件系统通常存放于内存和 Flash 中，或是基于网络的文件系统。根文件系统中存放了嵌入式系统使用的所有应用程序、库以及其他需要用到的服务。下图列出了根文件系统的顶层目录。

```
.                    //根目录
├─── bin             //基本命令的可执行文件
├─── dev             //设备文件
├─── etc             //系统配置文件，包括启动文件
├─── lib             //基本库文件，例如 C 库和内核模块 ko
├─── mnt             //临时文件系统挂载点
├─── opt             //添加的软件包
├─── proc            //内核以及进程信息的虚拟文件系统
├─── root            //root 用户目录
├─── sbin            //系统管理的可执行程序
├─── sys             //系统设备和文件层次结构
├─── system          //可读写系统文件(jffs2)
├─── tmp             //临时文件
├─── usr             //包含一些有用的文件
└─── var             //存放系统日志或一些服务程序的临时文件
```

4.2 文件系统的选择

文件系统有 glibc 与 uclibc 两种选择，glibc 的特点是支持功能全面，但是占用存储稍多；uclibc 的特点是占用存储空间小于 glibc，但是支持功能比 glibc 稍少。开发者应该根据自己实际情况选用适合的 libc 方式。

对于一般的应用场景，推荐客户使用 uclibc 搭建系统。

```
isvp_uclibc_defconfig                --uclibc 默认 config
isvp_uclibc_mini_defconfig           --uclibc 默认裁剪 config
isvp_glibc_defconfig                 --glibc 默认 config
isvp_glibc_mini_defconfig            --glibc 默认裁剪 config
```

对于 NorFlash Based 系统，存储空间一般较小，因此会选用压缩文件系统。推荐以下两种文件系统：

```
A. squashfs：只读文件系统，压缩率高
B. jffs2：可读写文件系统，可选择压缩方式
```

推荐的系统搭建的方案是----系统 rootfs 以及不需要经常修改的系统分区采用 squashfs 文件系统，而配置分区 system 等需要经常读写的分区采用 jffs2 文件系统。

4.3 文件系统的制作

关于文件系统制作的介绍与使用请参见《 T31 文件系统制作指南 》。

4.4 Demo rootfs 简单说明

ISVP 的 Demo 分区方式为：

```
A. u-boot:256K
B. uImage:2560K
C. rootfs:2048K
D. system:3328K
```

在 ISVP 的文件系统中，有两个指定的文件路径：

```
/etc/sensor/ ——此目录下存放 Sensor 的效果 bin 文件
/etc/webrtc_profile.init ——回音消除参数文件
```

不同的产品以上两处可能会有不同，而且产品升级时也可能升级相关的参数文件，因此建议将以上两处做软链接到可读写或者可更新的路径下。

Demo 的 rootfs 在启动后，会探测 system 分区是否可用，如果不可用，会进行格式化，并创建相关的目录结构。因此，第一次制作 Demo 系统，可以将整片 Flash 擦除，并烧录 u-boot， uImage，rootfs 即可；system 分区会自动创建。

system 分区的结构如下：

```
system
├─── bin   --此目录下放置应用程序
├─── etc
│     └─── sensor   --/etc/sensor 软链接到这里
│           └─── jxh42.bin  --效果文件
├─── init
│     └─── app_init.sh    --开机执行初始化脚本
└─── lib
      ├─── firmware   --/lib/firmware 软链接到这里
      └─── modules    --/lib/modules 软链接到这里
```

```
├──── sensor_jxh42.ko   --sensor 驱动
└──── tx-isp.ko    --ISP 驱动
```

注：Demo rootfs 可作为参考，开发者可以根据产品的实际情况定义 rootfs 方案。

5 系统启动与烧录

一个全新的板子里面什么都没有(准确的说芯片里面有固化的 Bootrom)，只是一个冷冰冰的机器，怎么让这个机器变得有灵魂呢？看下面介绍。

5.1 Uboot 启动

5.1.1 SD 卡启动

SD 卡插上读卡器就可以在 PC 上操作，给 SD 卡分出一个新分区，把编译好的 uboot 拷贝到这个分区的确定位置上，然后指导 Bootloader 指向 SD 卡的这个位置，启动 uboot (SD 卡启动卡制作查看 附录一)。

5.1.2 Nor/Nand 启动

nor/nand 是板子上的存储介质，本质上和 SD 卡里的存储介质类似，但这些存储器是焊接到板子上的，不能直接拿下来操作，而这些闪存是 EEPROM 掉电不丢失数据，所以把 uboot 拷贝到 nor/nand 闪存中，重启既不需要再次烧录。拷贝的方法有两种：一是 SD 卡启动 uboot 后，把编译好的 uboot.bin 文件烧录到 nor/nand flash 中；二是在焊接 flash 前，用烧录器把 uboot.bin 烧录到相应 flash 中。

5.2 系统烧录

5.2.1 tftpboot 传输与烧录

1) 目标

将文件通过 tftpboot 方式，从 PC 端，下载到 Uboot 的内存中。

2) 前提

硬件：开发板上网卡；板子通过网线连接到路由器或者交换机上，PC 也连到该路由器或者交换机上，并且处于同一网段上；

软件：PC 端安装并设置好 tftpboot 服务，把相应的 u-boot 等文件放到 tftpboot

根目录下；uboot 中支持网卡驱动并且支持相应的 tftpboot 操作。

3) 操作

在 uboot 中，执行 `tftpboot mem_addr file_name`；可以将文件 file_name 传送到 Uboot 的内存地址 mem_addr 中。

4) 传输实例

通过 tftpboot load 文件到内存；

```
$ mw.b  0x80600000 0xff 0x1000000
$ tftpboot 0x80600000 u-boot-with-spl.bin
$ tftpboot 0x80640000 uImage
$ tftpboot 0x808c0000 root-uclibc-1.1.squashfs
```

分别把 u-boot-with-spl.bin、uImage、root-uclibc-1.1.squashfs 下载到内存的 0x80600000、0x80640000、0x808c0000 处。

5) 烧录到 Nor

把下载到内存上的文件，写到开发板上的 nor flash 上，重启后可以从 nor 上直接读取文件，而不必每次手动 load 文件到内存上。烧录命令如下：

```
$ sf probe;sf erase 0x0 0x1000000;sf write 0x80600000 0x0 0x1000000
```

- `sf probe` 在使用 sf read、sf write 之前,一定要调用 sf probe
- `sf erase` 擦除指定位置，指定长度的 flash 内容，擦除后内容全 1；
- `sf write` 写内存数据到 flash 上。

5.2.2 SD 卡传输与烧录

1) 目标

将某个文件写到内存中，然后写到 flash。

2) 前提

硬件：开发板具备 SD 卡槽；使用读卡器从 PC 上拷贝数据到 SD 卡。

软件： uboot 支持 SD 卡驱动。

3) 操作

在 uboot 中，执行 `$ fatls mmc 0` 查看 SD 卡中的文件

`$ fatload mmc 0 mem_addr file_name` 将文件 file_name 传送到 Uboot 的内存地址 mem_addr 中。

4) 传输实例

擦除内存上的分区信息：

```
$ mw.b 0x80600000 0xff 0x1000000
```

fatload 命令 load 文件到内存；

```
$ fatload mmc 0 0x80600000 u-boot-with-spl.bin
$ fatload mmc 0 0x80640000 uImage
$ fatload mmc 0 0x808c0000 root-uclibc-1.1.squashfs
```

5) 烧录到 Nor

把下载到内存上的文件，写到开发板的 nor flash 上，重启后可以直接从 nor 上读取文件，而不必每次手动 load 文件到内存上。烧录命令如下：

```
$ sf probe;sf erase 0x0 0x1000000;sf write 0x80600000 0x0 0x1000000
```

- `sf probe` 在使用 sf read、sf write 之前,一定要调用 sf probe。
- `sf erase` 擦除指定位置，指定长度的 flash 内容，擦除后内容全 1。
- `sf write` 写内存数据到 flash 上。

5.2.3 USB 烧录

1) 目标
将某个文件写到 flash 上。
2) 前提
硬件：开发板上有 USB 烧录接口；数据线。
软件： USB 烧录 PC 工具。
3) 操作
参考《USBCloner 烧录指南.pdf》。

5.2.4 串口连接

开发板串口负责交互信息传输，内核打印会通过串口发送到显示屏上，开发板通过串口接收键盘的输入信息。

PC 与开发板串口的连接，需要配置对应的参数，如串口号，波特率，奇偶校验，数据位，停止位等。比如配置串口号为 com6，波特率为 115200，8 位数据位，一位停止位，无奇偶校验。

5.3 Uboot 配置

uboot 最终目的是启动加载内核。内核的启动需要 uboot 的引导并把相应的环境变量传到 kernel。此外还需要挂载一个根文件系统，存放内核配置信息和命令操作。

5.3.1 Uboot 命令学习

- reset：重新启动嵌入式系统。
- printenv：打印当前系统环境变量。
- setenv：设置环境变量，格式：`setenv name value`，表示将 name 变量设置成 value 值；如果没有这个参数，表示删除该变量。
- saveenv：保存环境变量到 flash 中。
- sleep：延迟执行，格式：`sleep N`，可以延迟 N 秒钟执行。
- run：执行环境变量中的命令，格式：`run var_name`，可以跟几个环境变量名。
- cp：在内存中复制数据块，格式：`cp source target count`，第一个参数是源地址，第二个参数是目的地址，第三个参数是复制数目。
- tftpboot：通过 tftpboot 协议下载文件到内存，格式 `tftpboot target source`
- fatls：显示 SD 卡的文件，格式 `fatls mmc 0`

- bootm：可以引导启动存储在内存中的程序映像。格式：`bootm addr1 addr2`，第一个参数是程序映像的地址，第二个参数一般是 RAMDISK 地址。

详细介绍可以在 uboot 命令行输入 help 查看。

5.3.2 Uboot 环境变量

板子上电启动，进入 uboot 命令行（读秒时按下 Enter 键）；通过 uboot 命令 printenv 可以查看 uboot 的环境变量：

```
isvp_t31# printenv
bootargs=console=ttyS1,115200n8 32M@0x0 rmem=32M@0x2000000 init=/linuxrc
rootfstype=squashfs root=/dev/mtdblock2 rw
mtdparts=jz_sfc:256k(boot),2560k(kernel),2048k(root),-(appfs)
bootcmd=sf probe;sf read 0x80600000 0x40000 0x280000; bootm 0x80600000
bootdelay=1
ipaddr=193.169.4.44
serverip=193.169.4.2
```

- `bootargs` 内核通过 bootargs 找到文件系统。
- `console=ttyS1,115200n8` 设置串口号为 ttyS1，波特率为 115200。
- `mem=32M@0x0 rmem=32M@0x2000000` 分配内存。
- `init=/linuxrc` 系统首先运行/linuxrc 文件。
- `rootfstype=squashfs` 使用压缩只读文件系统 squashfs。
- `root=/dev/mtdblock2 rw` 告诉内核根文件从 mtd 分区 3 挂载，可读写。
- `mtdparts=jz_sfc:256k(boot),2560k(kernel),2048k(root),-(appfs)` mtd 分区大小。
- `bootcmd` 启动命令。
- `sf probe;sf read 0x80600000 0x40000 0x280000; bootm 0x80600000` 从 nor 启。
- `ipaddr 193.169.4.80` 板子的 IP 地址。
- `serverip=193.169.4.2` 服务器 IP 地址 。

5.3.3 更改 mtd 分区大小

mtd 分区大小，在 uboot 中以命令行参数的形式传入 kernel，在 include/configs/isvp_t31.h 中，对于 8M Flash，找到 CONFIG_SFC_NOR，参考平台定义其中：`mtdparts=jz_sfc:256k(boot),2560k(kernel),2048k(rootfs),-(appfs)`就是 MTD 的分区情况，共 4 块分区，uboot 256k，kernel 2.5M，rootfs(squashfs)2M。-(appfs)具有自适应功能，当 Flash 为 8M 时，appfs 分区会自动分配成 3.25M；当 Flash 为 16M 时会自动分配成 11.25M。如果想修改，请保持总大小不变的情况下，按如上格式修改。一般 uboot、kernel、 rootfs 大小固定不变，建议用户不要修改，可以调整 appfs 分区大小。

5.3.4 更改 rmem 内存大小

rmem 大小同样是在 include/configs/isvp_t31.h 中修改，参考平台定义如下：

```
#define BOOTARGS_COMMON "console=ttyS1,115200n8 mem=32M@0x0
```

```
rmem=32M@0x2000000"
```

其中，`mem=32M@0x0 rmem=32M@0x2000000`就是对 mem 的相关设置；

`mem=32M@0x0`从 0 地址开始给系统分配 32M 内存；

`rmem=32M@0x2000000`从 0x2000000（32M）地址开始给 SDK 分配 32M；

T31 内嵌一块 64M 的内存，如要修改 rmem，请在总大小 64M 不变情况下按格式进行修改。示例：

假设 rmem 减小 4M，mem 增大 4M。

如果想减小 rmem，那么 mem 应该增大。所以 mem 大小为 36M，从 0 地址开始；rmem 为 28M，地址应该从 36M 开始。

```
    #define BOOTARGS_COMMON "console=ttyS1,115200n8 mem=36M@0x0 rmem=28M@0x
2400000"
```

6 系统资源使用与调试

6.1 ISP_Sensor

ISP 通过一系列算法完成对数字图像的效果处理。主要包括 3A、坏点校正、去噪、强光抑制、背光补偿、色彩增强、镜头阴影校正等处理。

6.1.1 ISP 驱动

1) 进入驱动文件夹 /drivers/isp-t31/tx-isp-t31；首先修改 Makefile 中 `ISVP_ENV_KERNEL_DIR` 宏定义，使之能够索引到正确的 kernel 路径。

2) 然后进行执行 `make clean; make`

3) 最后生成的 tx-isp-t31.ko 拷贝到系统中。

4) ISP 驱动注册时提供 module_param 参数有 isp_clk 参数，其参考配置：960p@30fps 使用 60Mhz，1080p@30fps 使用 80Mhz，3M@25fps 使用 125Mhz。例如 3M@25fps 使用 125Mhz，

```
$ insmod tx-isp-t31.ko isp_clk=125000000
```

6.1.2 Sensor 驱动

1) sensor 驱动依赖于 kernel 和 ISP 驱动，所以在编译 sensor 驱动之前 kernel 和 ISP 驱动必须已经编译完成。

2) 进入具体的 sensor 驱动文件夹，首先修改 Makefile 中 `ISVP_ENV_KERNEL_DIR` 和 `ISP_DRIVER_DIR` 宏定义，使之能够索引到正确的 kernel 和 ISP 驱动路径。

3) 然后进行执行 `make clean;make`

4) 最后将生成的 sensor_xxx.ko 拷贝到系统中。

5) sensor 驱动注册时提供了多个 module_param 可配置参数，如果产品硬件遵循参考设计 insmod 时无需跟参数，使用默认值即可。

A. reset，power_down 的配置：

```
$ insmod sensor_xx.ko reset_gpio=18 pwdn_gpio=20
```

其中，gpio 的值为 GPIO 编号，规则为：num = 32 * n + bit，例如：PA18 的 GPIO 编号为 18，PC20 的 GPIO 编号为 84。

B. DVP 数据端口配置：

```
$ insmod sensor_xx.ko sensor_gpio_func=1
```

其中 sensor_gpio_func 为 DVP Port 的配置选项，0：PA Low-10bit；1：PA High-10bit；

2：PA 12bit。

C. Sensor 的数据接口有多种，目前支持 DVP 以及 MIPI CSI-2 两种接口；但有的 senosr 可能同时支持两种数据接口，为了区分加入了模块参数 data_interface，例如：

```
$ insmod sensor_xx.ko data_interface=1
```

其中，data_inteface 为 sensor 数据接口配置选项 1：MIPI 2：DVP

D. 基于降低功耗的考量，在 sensor 驱动中加入了配置最高帧率的参数 sensor_max_fps。一般 sensor 驱动中支持 30/25fps 与 15fps 的切换。例如：

```
$ insmod sensor_xx.ko sensor_max_fps=15
```

其中，sensor_max_fps 为 sensor 帧率配置选项。15:配置为 15fps， 25: 配置为 25fps，默认为 30/25fps 模式。

6.1.3 Avpu 视频编码驱动

1) 进入驱动文件夹/drivers/avpu/；首先修改 Makefile 中 `ISVP_ENV_KERNEL_DIR` 宏定义，使之能够索引到正确的 kernel 路径，视频编码驱动只依赖内核，所以在编译 avpu 驱动之前 kernel 必须已经编译完成；

2) 然后进行执行 `make clean;make`

3) 最后生成的 avpu.ko 拷贝到系统中。

4) avpu 驱动注册时提供了多个 module_param 可配置参数，如果产品硬件遵循参考设计 insmod 时无需跟参数，使用默认值即可。例如：

```
$ insmod avpu.ko clk_name='vpll' avpu_clk=400000000
```

其中 clk_name 是 vpu 选择使用的时钟源，avpu_clk 用于设置 avpu 的时钟频率。

6.1.4 Sensor 探测识别驱动

使用同一个开发平台支持多个 sensor 的情况。可以调用该驱动先探测 sensor 型号，加载正确的 sensor 驱动进行后续操作。

Sensor 识别驱动位于 drivers/misc/sensor_info 目录下，用户可以通过 ioctl 或者 proc 接口进行 Sensor 型号的查询。使用方法可参考例子 sample_sinfo.c。

6.1.5 图像效果文件

不同的 Sensor 以及镜头可能需要不同的效果参数配置，配置文件位置在：/etc/sensor 目录下，文件名为[sensor]-t31.bin，例如：ov9732-t31.bin。如果没有这个文件图像颜色等可能不正常。实际产品中效果 bin 文件往往需要随版本迭代更新，因此/etc/sensor 目录需要有读写权限。可供参考的方法之一是将/etc/sensor 目录做成一个软链接，链接到一个 rw 的分区中，这样就可以在版本更新时单独更新 bin 文件。

6.1.6 Sample 的使用

Sample 程序位于/samples/libimp-samples，里面是依赖 SDK 库的应用程序，包括图片的抓取、图片格式的转换、智能检测、录音放音、回应消除等应用程序。设置好 Makefile，直接在 sdk 目录下 make 即可以编译。用户可以参考提供的 sample 程序，编

写自己相应工程代码。

完成编译后，可以在目录下找到对应的可执行程序，拷贝到开发板上测试即可。详细操作查看下表；

表 6-1 sample 功能

应用文件	功能	执行命令	执行结果
sample-Ai.c	录音	./sample-Ai	生成录音文件 ai_record.pcm
sample-Ao.c	音频文件播放	./sample-Ao	播放采样率 16K 的音频文件 ao_paly.pcm
sample-Ai-AEC.c	音频回声消除	./sample-Ai-AEC	生成 test_for_play.pcm 和 test_aec_record.pcm 两个录音文件
sample-Ai-Ref.c	验证音频获取回声消除算法参考帧	./sample-Ai-Ref	生成 test_for_play.pcm 和 test_aec_record.pcm 两个录音文件
sample-dmic.c	数字麦克阵列录音	./sample-dmic	录音生成 dmic0_record.pcm，dmic1_record.pcm，分别对应麦克阵列中的 MIC0，MIC1 录制的声音数据
sample-Capture-nv12.c	抓取 NV12 图片	./sample-Capture-nv12	在/tmp 下面产生抓取 NV12 的图片
sample-Decoder-jpeg.c	把 jpeg 解码成 yuv	./ sample-Decoder-jpeg	在/tmp 下面生成 yuv 文件
sample-Encoder-jpeg.c	把 NV12 图片编码成 jpeg 图片	./ sample-Encoder-jpeg	将 NV12 图片编码成 jpeg 图片，并保存在/tmp 下
sample-Framesource.c	保存对应格式的视频	./ sample-Framesource	保存对应格式的视频（nv12 ,bgr0,raw）

6.2 Watchdog

1) 内核配置

在内核源码根目录下执行 make menuconfig 命令进入配置界面，

按下面方式配置

```
Symbol: JZ_WDT [=y]
```

```
Type  : tristate
Prompt: Ingenic jz SoC hardware watchdog
    Location:
      -> Device Drivers
        -> Watchdog Timer Support (WATCHDOG [=y])
```

2) 应用例程参考

如果客户使用 Watchdog，请参考：

Ingenic-SDK/samples/libsysutils-samples/sample-wdt.c

用户常遇到的问题，没有调用 wdt_disable()；就直接 close()；这样会导致关不掉从而出现错误！

6.3 GPIO

GPIO 是“General Purpose Input/Output”的简称，具有输入输出，功能复用的功能。开发者在开发硬件驱动时往往需要操作 GPIO，以控制外设硬件。

ISVP 使用 Linux 标准的 GPIOLIB 接口。GPIOLIB 提供了统一的 GPIO 申请/释放/设置/获取接口，按照 GPIOLIB 的设定，需要在 Kernel space 进行调用。如果是 User space 需要操作 GPIO，有两种方法可以选择：

A. 通过 sysfs GPIO 接口进行操作

B. 应用程序调用相关的驱动，驱动中实现 GPIO 的设置

6.3.1 头文件及 API

GPIOLIB 的头文件为：include/linux/gpio.h

在驱动程序中加入头文件引用：

```
#include <linux/gpio.h>
```

API 在头文件 include/asm-generic/gpio.h 中定义，例如：

```
int gpio_request(unsigned gpio, const char *label);
void gpio_free(unsigned gpio);
int gpio_direction_input(unsigned gpio);
int gpio_direction_output(unsigned gpio, int value);
int gpio_get_value(unsigned int gpio);
void gpio_set_value(unsigned int gpio, int value);
int gpio_to_irq(unsigned int gpio);
```

详细的文档说明可参考 kernel/Documentation/gpio.txt；参考代码 Linux kernel 中标准驱动的 GPIO 操作均使用标准的 GPIOLIB，比如 Software I2C，Fixed regulator，以及中断等。

6.3.2 sysfs GPIO

sysfs GPIO 是 Linux 标准的用户空间操作 GPIO 的接口。用户可通过命令行或者应用程序直接设置 GPIO 的输入/输出，高低电平等属性。一般情况下，GPIO 调试或者简单的 GPIO 应用（比如 IR-Cut 操作），可通过 sysfs GPIO 接口进行快速开发。

6.3.2.1 内核选项

在内核源码根目录下执行$ make menuconfig 命令进入配置界面，选中以下选项：

```
Device Drivers  --->
      -*- GPIO Support  --->
          [*]   /sys/class/gpio/... (sysfs interface)
```

一般情况下，内核的默认配置已经勾选了此选项。

6.3.2.2 sysfs GPIO 的申请与释放

在操作 sysfs GPIO 之前需要对其进行申请。值得注意的是，由于申请 sysfs GPIO 会在内核 request_gpio，因此在内核中已经申请过的 GPIO 在 sysfs GPIO 再次申请会失败。

申请/释放 GPIO 方法如下：

```
$ cd /sys/class/gpio
$ echo [gpio_num] > export         #申请 GPIO
$ echo [gpio_num] > unexport       #释放 GPIO
```

注：gpio_num 即 GPIO 号。计算公式为：

PA(n) = 0 * 32 + n

PB(n) = 1 * 32 + n

PC(n) = 2 * 32 + n

...

例如：申请 PB(10) = 1 * 32 + 10 = 42

```
$ echo 42 > export
```

申请后在/sys/class/gpio 目录下即会出现 gpio42 目录。

```
$ echo 42 > unexport
```

释放后 gpio42 目录也会消失。释放后的 GPIO 状态并不会恢复，会保持申请时的状态（电平等）。

6.3.2.3 设置输入/输出方式

在申请 GPIO 后，进入 gpioN 目录，例如 gpio42，进行如下操作：

```
$ echo out > direction             #设置 PB10 为输出模式
$ echo in > direction              #设置 PB10 为输入模式
```

6.3.2.4 设置有效电平

gpioN 目录下有 active_low 节点，表示当前 GPIO 的有限电平，默认为 0，其意义为，当输入/输出 value 为 0 时，GPIO 为低电平，当输入/输出 value 为 1 时，GPIO 为高电平。同样的，当 active_low 为 1 时，当输入/输出 value 为 0 时，GPIO 为高电平；当输入/输出 value 为 1 时，GPIO 为低电平。

也就是说，GPIO 的真实电平=value^active_low。

```
$ echo 0 > active_low    #value 是 0,表示低电平。value 是 1,表示高电平
```

```
$ echo 1 > active_low    #value 是 1,表示低电平。value 是 0,表示高电平
```

6.3.2.5 输入/输出

gpioN 目录下有 value 节点，表示 gpioN 的电平：当 GPIO 为输入模式时，读取到 value 的值异或 active_low 即为 GPIO 的电平；当 GPIO 为输出模式时，写入到 value 的值异或 active_low 即为 GPIO 的输出电平。

```
$ cat value                #读取电平（输入模式）
$ echo 0 > value           #设置电平（输出模式）
```

6.4 Exfat

1) 内核选项

如果客户需要支持 exfat 文件系统，编译加载 drivers/misc/exfat-nofuse 下的驱动即可，使用默认的内核配置。

不提供设备上格式化 exfat 的工具。

6.5 ADC

1) 内核配置

在内核源码根目录下执行 make menuconfig 命令进入配置界面，

按下面方式配置

```
Device Drivers  --->
    Multifunction device drivers  --->
        <*> Support for the XBurst SADC core
        <*> Support for the XBurst SADC AUX
        <*> Support for XBurst TCU
```

2) 应用例程参考

参考代码在 sample_adc.c

T31 ADC 的基准电压为 1.8V。

注意：参考代码中 ADC_PATH 表示选择哪个 ADC；STD_VAL_VOLATAGE 表示参考电压，单位为 mv！

6.6 Motor

6.6.1 驱动介绍

电机的 sample 驱动是使用 GPIO 和定时器来控制四相八拍步进电机的转动。驱动支持两个电机同时使用。参考代码在 drivers/misc/下面；sample_motor 目录为不限位开关的电机驱动；sample_motor2 为带有限位开关的电机驱动；sample_motor3 为带细

分器的电机驱动。

电机的控制单位是"步"。在实际产品中，两个电机受到结构的限制有最大转动角度。驱动中定义了两个坐标点(0,0),(vmaxstep,hmaxstep)分别对应两个电机转动的结构限制点。vmaxstep 和 hmaxstep 两个参数可以在加载电机驱动时以参数的形式设置，跟产品的结构转动限制和齿轮转速比有关。

6.6.2 接口介绍

表 6-2 Motor API

函数	功能
MOTOR_RESET	RESET 是使用电机前必须设置的。RESET 操作会控制电机先运动到(0，0)坐标点，然后运动到(vmaxstep/2，hmaxstep/2)坐标中心点。上电前，每个电机所处结构的位置可能都不一样，RESET 操作就是把两个电机都初始化到(vmaxstep/2，hmaxstep/2)坐标中心点。
MOTOR_MOVE	MOVE 操作是控制两个电机转动一定的步数，参数 x，y 对应两个方向，是相对坐标。
MOTOR_GET_STATUS	GET_STATUS 接口可以得到目前电机的状态和坐标，这个坐标是绝对坐标。
MOTOR_SPEED	电机的转速范围是 100-900。值越大，速度越快。一般设置 400/500。
MOTOR_GOBACK	控制电机转动到(vmaxstep/2，hmaxstep/2)坐标中心点。
MOTOR_STOP	STOP 接口控制电机停止。
MOTOR_CRUIS	CRUISE 接口控制两个电机以最大行程巡航。

6.6.3 驱动适配

根据硬件设计，驱动需要修改 GPIO 配置。在文件 motor.h 中。例如：

```
#define HORIZONTAL_ST1_GPIO     GPIO_PB(22) /**< Phase A */
#define HORIZONTAL_ST2_GPIO     GPIO_PB(21) /**< Phase B */
#define HORIZONTAL_ST3_GPIO     GPIO_PB(20) /**< Phase C */
#define HORIZONTAL_ST4_GPIO     GPIO_PB(19) /**< Phase D */

#define VERTICAL_ST1_GPIO       GPIO_PC(7)
#define VERTICAL_ST2_GPIO       GPIO_PC(6)
#define VERTICAL_ST3_GPIO       GPIO_PC(5)
#define VERTICAL_ST4_GPIO       GPIO_PC(4)
```

注意：产品使用的 gpio 不一样，这点需要认真对比。驱动使用 TCU 定时器，如果客户使用 PWM 驱动，请确认 PWM 驱动使用的 TCU 通道，以防和电机使用的 TCU 产生冲突！

6.7 PWM

1) 内核配置

在内核源码根目录下执行 make menuconfig 命令进入配置界面，不使用内核自带的驱动程序。

```
Device Drivers  --->
    [*] Pulse-Width Modulation (PWM) Support  --->
        [*]   JZ PWM driver (NEW)
              *** JZ PWM function pin select ***
```

2) 驱动参考

参考代码 drivers/misc/sample_pwm 中。本章节适用的驱动版本为“H20180309a”

注意点：使用相关 PWM 时会使用到 TCU 模块；如果产品同时使用电机，请确认是否和电机驱动中的 TCU 冲突！

设置的 polarity 表示有效电平的极性，duty 表示有效电平的占空。例如：polarity = 0，duty=0：表示低电平为有效电平，同时有效电平的占空为 0，此时应该输出的是高电平。

客户参考代码为 test_pwm.c；其中设置周期的单位为 纳秒！

6.8 SPI

SPI, Serial Perripheral Interface, 串行外围设备接口，它允许 MCU 以全双工的同步串行方式, 与各种外围设备进行高速数据通信。

6.8.1 内核配置

在内核源码根目录下执行 make menuconfig 命令后进入配置界面，

按下面方式配置：

```
Device Drivers  --->
    [*] SPI support  --->
        <*>   Ingenic JZ series SPI driver
        [*]    Ingenic SoC SSI controller 0 for SPI Host driver
        -*-      JZ SSI0 controller function pins select
```

6.8.2 添加一个 SPI Flash

uboot 和 kernel 需要同步添加，

1) 对于 uboot

打开 drivers/spi/jz_spi.h ；找到 jz_spi_support_table 数组，把新加的 flash 按格式放进去，可以 copy 一份任意现有的 flash 配置，主要修改 name、id_manufactory、sector_size、size，sector_size 是 flash 的擦大小，建议设成 64K，提高擦除速度，如果不支持 64K 请设成 32K。

以添加一个 8M flash EN25QH64 为例：

```
{
```

```
        .name           = “EN25QH64”,
        .id_manufactory = 0x1c7017,
        .page_size       = 256,
        .sector_size      = (32 * 1024),
        .addr_size = 3,
        .size           = (8 * 1024 * 1024),
        .quad_mode = {
            .dummy_byte = 8,
            .RDSR_CMD = CMD_RDSR_1,
            .WRSR_CMD = CMD_WRSR_1,
            .RDSR_DATE = 0x2,//the data is write the spi status register for
QE bit
            .RD_DATE_SIZE = 1,.WRSR_DATE = 0x2,//this bit should be the flash
QUAD mode enable
            .WD_DATE_SIZE = 1,
            .cmd_read = CMD_QUAD_READ,
    #ifdef CONFIG_JZ_SFC
            .sfc_mode = TRAN_SPI_QUAD,
    #endif
        },
    },
```

如果要使用四线模式，请根据 spec 定义设置 quad_mode 参数。

还需要查看 drivers/mtd/spi/spi_flash.c 中关于 flash 厂商号是否存在，如果不存在需要自己添加。

2) 对于 kernel

```
$ vi arch/mips/xburst/soc-t31/chip-t31/isvp/common/spi_bus.c
```

找到 spi_nor_pdata 定义添加方法与 uboot 类似，只是参数名称不太一样，其中 id 对应 uboot 中 id_manufactory，erasesize 对应 uboot 中 sector_size，chipsize 对应 uboot 中 size，pagesieze 和 sectorsize 可以不改。

```
    {
        .name           = “EN25QH64”,
        .pagesize       = 256,
        .sectorsize     = 4 * 1024,
        .chipsize       = 8192 * 1024,
        .erasesize      = 32 * 1024,//4 * 1024,
        .id             = 0x1c7017,
        .block_info     = flash_block_info,
        .num_block_info = ARRAY_SIZE(flash_block_info),
        .addrsize       = 3,
        .pp_maxbusy     = 3,            /* 3ms */
        .se_maxbusy     = 400,          /* 400ms */
        .ce_maxbusy     = 8 * 10000,    /* 80s */
```

```
    .st_regnum      = 3,
#ifdef CONFIG_SPI_QUAD
    .quad_mode = &flash_quad_mode[0],
#endif
},
```

6.9 I2C

1) 内核配置

在内核源码根目录下执行 make menuconfig 命令进入配置界面，选中以下选项：

```
Device Drivers  --->
        <*> I2C support  --->
            [*]   Autoselect pertinent helper modules
                  I2C Hardware Bus support  --->
                    <*> JZ_v12 i2c controler 0 Interface support
                        JZ_v12 i2c controler 0 function
                        pins select (GPIO_PA12 & GPIO_PA13)  --->
```

根据需求选择硬件 i2c 或者软件 i2c；硬件 i2c 是确定的引脚，软件 i2c gpio 可以在 /kernel/arch/mips/xburst/soc-t31/chip-t31/isvp/Swan/board.h 中修改。

2) 驱动编译

在/opensource/drivers/i2c/sample_eeprom 目录下有使用内核 i2c 驱动的 at24.c 例子，在编译直接需要确认 Makefile 中的 kernel 路径正确性。可以实现 i2c 读写内存。

6.10 SLCD

1) 内核配置

```
Device Drivers  --->
    <*> Multimedia support  --->
        Graphics support  --->
           <*> Support for frame buffer devices  --->
```

2) 驱动编译

驱动代码位于 drivers/sample_slcd；在编译前需要确认 Makefile 中的 kernel 路径正确性和 lcd_device 目录中的初始化 GPIO 需要和硬件对应。

/lcd-device_truly240320.c

```
#define GPIO_LCD_CS           GPIO_PB(20)
#define GPIO_LCD_RD           GPIO_PB(19)
#define GPIO_BL_PWR_EN        GPIO_PC(20)
#define GPIO_LCD_RST          GPIO_PB(22)
```

/lcd-driver_truly240320.c

```
jzgpio_set_func(GPIO_PORT_B, GPIO_FUNC_3, 0x3f<<6 | 0x3<<13 | 0x3<<15 | 0x7<<20);
```

注意：T21 平台上 SLCD 使用的时钟是 AHB1，默认没打开；T31 上使用的时钟是 AHB0，默认已经打开。

6.11 Audio

音频分为内部 codec 和外部 codec，内部 codec 又分数字 mic 和模拟 mic。

6.11.1 内核配置

在内核源码根目录下执行 make menuconfig 命令进入配置界面，按需求进行下面配置。

1) Amic 配置(默认已配置)

```
Device Drivers --->
      <*>Sound card support --->
           <*>Open Sound System --->
              [*]Open Sound System of Xburst --->
                [*]Compile jzsound driver into ko
                [*] Jz On-Chip I2S driver
                [*] xburst internal coedc --->
                       [*] t10 internal codec
```

2) Dmic 配置

```
Device Drivers --->
      <*>Sound card support --->
          <*>Open Sound System --->
             [*]Open Sound System of Xburst --->
                [*]Compile jzsound driver into ko
                [*] Jz On-Chip I2S driver
                [*] xburst internal coedc --->
                    [*] t10 internal codec
                [*] jz soc t31 dmic enable
```

3) 外部 Codec ES8374 配置

```
Device Drivers --->
      <*>Sound card support --->
           <*>Open Sound System --->
              [*]Open Sound System of Xburst --->
              [*]Compile jzsound driver into ko
              [*] Jz On-Chip I2S driver
              [*] xburst internal coedc --->
```

```
                    [*] t10 internal codec
                [*] jz soc t31 dmic enable
                [*] xburst external codec --->
                     <*> JZ_v12 i2c controler 2 Interface support
```

6.11.2 Audio 驱动

驱动代码位于/opensource/drivers/audio/oss2；在编译前需要确认 Makefile 中的 kernel 路径正确性。驱动参数使用如下。

1) Audio 驱动加载时提供了参数 spk_gpio 用于配置功放的使能，驱动默认配置的是 PB31；如果不需要驱动控制该引脚，配置该参数为 spk_gpio=-1。

```
# insmod audio.ko spk_gpio=-1
```

2) Audio 支持同时使用 Amic 和 Dmic 功能；默认关闭。如果使用需要内核同时配置上 Amic 和 Dmic。Audio 同时使用 Amic 和 Dmic 时，可以通过设置参数 dmic_amic_sync=1 来确保 amic 和 dmic 的数据时间差固定。

```
# insmod audio.ko dmic_amic_sync=1
```

3) Codec 支持输入 ALC 功能，用于自动控制输入增益；默认开启。驱动加载时提供参数 alc_mode 设置 ALC 开启和关闭。该参数配置成为 0 时，关闭输入 ALC 功能，使用手动模式配置输入增益。

ALC 开启后，使用 IMP_AI_SetAlcGain 接口设置输入相对增益。ALC 关闭后，使用 IMP_AI_SetGain 接口手动设置输入增益。

```
# insmod audio.ko alc_mode=0
```

6.12 Wifi

目前 wifi 模组主要存在两种接口方式：USB 和 SDIO；T 系列芯片两种接口都支持。现在主流 wifi 芯片调试完成的型号如下：

A. USB 接口：MT7601，RTL8188，RTL8723BU，ATBM6022，RDA5995

B. SDIO 接口：AP6212，AP6212A，AP6181，RTL8189FS，RTL8189ES，SSV6x5x，Hi3881V100

6.12.1 Wifi 内核配置

对于 USB 接口的 wifi 首先需要配置内核 USB 模块：

```
Device Drivers--->
    [*] USB support  --->
        <*> Support for Host-side USB
            <*> DesignWare USB2 DRD Core Support
                Driver Mode (Host Mode Only)  --->
```

对于 SDIO 接口的 wifi 首先需要配置内核 MMC1 模块：

```
Device Drivers  --->
   <*> MMC/SD/SDIO card support  --->
      [*]     JZMMC_V12 MMC1
```

配置好了之后，对于上述的 wifi 模组，有的在内核中已经有驱动了，只需在 make menuconfig 中配置一下即可使用。有的需要使用 wifi 模组厂商提供的驱动源码包编译成 ko 文件才可以使用。

6.12.1.1 MT7601

```
Device Drivers--->
    [*] Network device support --->
        [*] Wireless LAN --->
            <*> MT7601_util STA support
```

6.12.1.2 RTL8188/RTL8723BU/ATBM6022/RDA5995

使用 wifi 模组厂商提供的驱动源码包。

6.12.1.3 AP6212/AP6212A

这两款 wifi 内核配置如下：

第一步：选择 BCM 驱动；

```
Device Drivers--->
     Misc devices --->
        <*> BCM module power control core driver
```

第二步：选择 AP6212，AP6212A 驱动

```
Device Drivers--->
  [*] Network device support --->
      [*] Wireless LAN --->
          <*> Broadcom bcm43438 wireless cards support
              Interrupt type (In-Band Interrupt)
```

第三步：选择 MMC1 的驱动

```
Device Drivers  --->
   <*> MMC/SD/SDIO card support  --->
       [*]     JZMMC_V12 MMC1
```

6.12.1.4 AP6181

第一步：选择 BCM 驱动；

```
Device Drivers--->
   Misc devices --->
        <*> BCM module power control core driver
```

第二步：选择 AP6181 驱动

```
Device Drivers--->
```

```
    [*] Network device support --->
        [*] Wireless LAN --->
            <*>Broadcom 43341 wireless cards support
```

第三步：选择 MMC1 的驱动

```
Device Drivers  --->
   <*> MMC/SD/SDIO card support  --->
       [*]     JZMMC_V12 MMC1
```

6.12.1.5 RTL8189FS/RTL8189ES/SSV6x5x/Hi3881V100

使用 wifi 模组厂商提供的驱动源码包。

6.12.2 Wifi 驱动编译

1) USB 接口的 wifi 驱动，只需要指定编译工具链和 kernel 路径，直接编译即可。

以 RTL8723BU 为例：打开 Makefile，参照添加 `CONFIG_PLATFORM_INGENIC = y`；然后参照现有板机继续添加：

```
1300 ifeq ($(CONFIG_PLATFORM_INGENIC), y)
1301 EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_MINIMAL_MEMORY_USAGE
1302 ARCH ?= mips
1303 CROSS_COMPILE ?= mips-linux-gnu-
1304 KSRC ?= /home_d/ywhan/work/isvp/opensource/kernel
1305
1306 ifeq ($(CONFIG_SDIO_HCI), y)
1307 EXTRA_CFLAGS += -DCONFIG_PLATFORM_OPS
1308 _PLATFORM_FILES += platform/platform_ingenic_sdio.o
1309 endif
1310 endif
```

2）SDIO 接口的 wifi 驱动，首先需要在 platform 文件夹下面添加 power_en 引脚控制代码 platform_ingenic_sdio.c，该文件主要定义了 WIFI 模组的 power_en 引脚的控制，代码默认引脚为 PB(30)。以 RTL8189ES 为例：

首先添加 platform_ingenic_sdio.c，确定硬件对应的 power_en 引脚和软件是否匹配。打开 Makefile，添加 `CONFIG_PLATFORM_INGENIC = y` 然后参照现有板机继续添加

```
1315 ifeq ($(CONFIG_PLATFORM_INGENIC), y)
1316 EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN -DCONFIG_MINIMAL_MEMORY_USAGE
1317 EXTRA_CFLAGS += -DCONFIG_IOCTL_CFG80211 -DRTW_USE_CFG80211_STA_EVENT
1318 ARCH ?= mips
1319 CROSS_COMPILE ?= mips-linux-gnu-
1320 KSRC ?= /home_d/ywhan/isvp/opensource/kernel
1321
1322 ifeq ($(CONFIG_SDIO_HCI), y)
1323 EXTRA_CFLAGS += -DCONFIG_PLATFORM_OPS
1324 _PLATFORM_FILES += platform/platform_ingenic_sdio.o
1325 endif
1326 endif
```

注：AP6212，AP6212A，AP6181 驱动已经集成到内核不需要单独编译。

6.12.3 Wifi 启动操作流程

加载 KO， 成功会生成 wlan0 节点，通过 ifconfig -a 可以看到。

第一步： `$ ifconfig wlan0 up` 启动 WIFI（可选操作）

第二步： `$ wpa_supplicant -D wext -i wlan0 -c wpa_supplicant.conf -B`

（wpa_supplicant 应用软件的参考附录）

第三步： 连接 WIFI：自动配置 IP；`$ udhcpc -i wlan0` 获取 IP 地址，网关等信息（注意 udhcpc.script 的路径匹配）。

或者手动配置 IP；

```
$ ifconfig wlan0 193.169.4.75
$ route add default gw 193.169.4.1
$ echo "nameserver 193.169.1.57" > /etc/resolv.conf
```

6.12.4 提高 SDIO wifi 工作时钟

针对需要提高 SDIO wifi 工作时钟的操作；一般使用默认的 24Mclk 即可满足工作，假如需要提高 wifi 的工作时钟，以 T31 芯片为例参考步骤如下：

第一步： 进入内核文件夹，打开 arch/mips/xburst/soc-t31/chip-t31/isvp/common/mmc.c ；修改 sdio_pdata 结构体中 capacity 变量，在原值基础上添加 MMC_CAP_MMC_HIGHSPEED。

第二步： 通过 make menuconfig 进入 MMC1 的配置，可以将频率提高到 36M，48M。

6.12.5 针对 BCM wifi 的额外说明

BCM WIFI 模块驱动在内核中实现，内核针对 BCM 模块封装了四个板级电源管理接口函数，

```
bcm_wlan_power_on(),
bcm_wlan_power_off(),
bcm_manual_detect(), bcm_customer_wlan_get_oob_irq();
```

用于实现对 WIFI 芯片的上电、下电，sdio 的侦测和复位操作，对应操作中使用的

GPIO 也做了宏定义，用户如有变动，只需要修改相应的宏定义即可。T31 芯片的 GPIO 定义在 arch/mips/xburst/soc_t31/chip_t31/isvp/Swan/board.h 中。

```
/* **************GPIO WIFI START*************** */
#define WL_WAKE_HOST    GPIO_PB(8)
#define WL_REG_EN   GPIO_PC(9)
#define WL_MMC_NUM  1 //sdio use MMC1
#define WLAN_PWR_EN (-1)
#define BCM_PWR_EN  (-1)
#define WL_32K_EN   GPIO_PB(18)
/* ***************GPIO WIFI END***************** */
```

6.13 USB

6.13.1 USB 工作模式配置

USB 设备模式和 USB 主机模式。在 USB 设备模式下，外部 USB 硬件充当 USB 主机，开发板上的 USB 属于从机设备；USB 主机模式正好相反，开发板属于主机，外部 USB 设备属于从机设备。

6.13.1.1 Device Only 模式

```
Device Drivers --->
      [*] USB support --->
          <*> DesignWare USB2 DRD Core Support
            Driver Mode(Device Mode Only) --->
```

6.13.1.2 Host Only 模式

```
Device Drivers --->
     [*] USB support --->
          <*> DesignWare USB2 DRD Core Support
             Driver Mode (Host Mode Only) --->
```

6.13.1.3 Both 模式

```
Device Drivers --->
      [*] USB support --->
          <*> DesignWare USB2 DRD Core Support
             Driver Mode (Both Host and Device) --->
```

6.13.1.4 模式的选择

设置为 both 模式时，host 与 devic 切换有两种方式：

A. 硬件使用 ID PIN 自动切换
B. 软件手动设置

```
//set host mode:
$ devmem 0x10000040 32 0x0b000096
// set device mode:
$ devmem 0x10000040 32 0x0b800096
$ devmem 0x13500000 32 0x001100cc
```

6.13.2 USB 接口配置

由于 USB 外设种类繁多，内核默认配置不能完全兼顾，用户可以根据需要自行添加。

6.13.2.1 U 盘配置

第一步：进入配置界面 make menuconfig 配置如下

```
a)   Device Driver --->
              SCSI device support --->
                  <*> SCSI device support
                  <*> SCSI disk support
b)     Device Driver --->
   [*] USB support  --->
       <*> Support for Host-side USB
       <*> USB Mass Storage support
       <*> DesignWare USB2 DRD Core Support
           Driver Mode (Host Mode Only)
```

第二步：编译内核 make uImage 并烧录到开发板上；

第三步：插上 U 盘，内核会打印相应提示信息。

6.13.2.2 USB 转串口

第一步：进入配置界面 make menuconfig 配置如下

```
a) Device Driver --->
[*] USB support  --->
<*> Support for Host-side USB
<*> USB Modem (CDC ACM) support
<*> USB Serial Converter support  --->
   <*> USB Prolific 2303 Single Port Serial Driver //选择串口芯片，2303 的芯片配置为例。
```

第二步：编译内核 make uImage 并烧录到开发板上；

第三步：插上串口线，内核会打印相应提示信息。

6.13.2.3 USB 硬盘的配置

第一步：进入配置界面 make menuconfig 配置如下

```
a) Device Driver --->
       SCSI device support --->
            <*> SCSI device support
            <*> SCSI disk support
            [*] SCSI low-level drivers (NEW)  --->
b) Device Driver --->
      [*] USB support  --->
           <*> Support for Host-side USB
           <*> USB Mass Storage support
           <*> DesignWare USB2 DRD Core Support
              Driver Mode (Host Mode Only)
c) [*] Enable the block layer  --->
      [*]  Support for large (2TB+) block devices and files
```

第二步：编译内核 make uImage 并烧录到开发板上；

第三步：连接硬盘，启动系统，内核会打印相应提示信息。

6.13.2.4 USB 以太网卡

第一步：进入配置界面 make menuconfig 配置如下

```
a) Eth Config
Device Drivers  --->
    [*] Network device support  --->
         USB Network Adapters  --->
            <*> Multi-purpose USB Networking Framework
            <*>   ASIX AX88xxx Based USB 2.0 Ethernet Adapters (NEW)
b) USB Mode Config
   Device Drivers  --->
       [*] USB support  --->
           <*> DesignWare USB2 DRD Core Support
               Driver Mode (Host Mode Only)  --->
```

第二步：编译内核 make uImage 并烧录到开发板上；

第三步：启动系统，查看相应信息。

6.13.2.5 USB 鼠标配置

第一步：进入配置界面 make menuconfig 配置如下：

```
a) USB Mode Config
    Device Driver --->
```

```
      [*] USB support   --->
        <*> Support for Host-side USB
        <*> DesignWare USB2 DRD Core Support
           Driver Mode (Host Mode Only)
b) Input system Config
    Device Driver --->
       Input device support  --->
         <*>   Mouse interface
            [*]   Mice  --->
c) USB His Config
    Device Drivers  --->
       HID support  --->
       -*- HID bus support
       /dev/hidraw raw HID device support
       USB HID support  --->
          <*> USB HID transport layer
          [*] /dev/hiddev raw HID device support
```

第二步：编译内核 make uImage 并烧录到开发板上

第三步：启动系统，插入鼠标，查看相应的信息。

```
[    23.101552] usb 1-1: new low-speed USB device number 2 using dwc2
[    23.313232] input: Logitech USB Optical Mouse as /devices/platform/jz-dwc2/dwc2/usb1/1-1/1-1:1.0/input/input1
[    23.329602] hid-generic 0003:046D:C05A.0001: input,hidraw0: USB HID v1.11 Mouse [Logitech USB Optical Mouse] on usb-dwc2-1/input0
```

我们 cat /dev/input/event1 这个节点，然后移动鼠标，在串口上可以看到一连串的打印（打印信息乱码显示，这些信息是鼠标的坐标信息等），停止移动鼠标就没有这些信息。

6.13.2.6 USB RNDIS 配置

USB RNDIS（远程网络驱动接口规范），产品为 USB 从设备。

第一步：进入配置界面 make menuconfig 配置如下

```
   a) Device Driver --->
        [*] USB support  --->
            <*> Support for Host-side USB
            <*> USB Gadget Support  --->
            <*>   USB Gadget Drivers (Ethernet Gadget
                (with CDC Ethernet support))  --->
            [*] RNDIS support (NEW)
   b)  Device Driver --->
         [*] USB support  --->
             <*> DesignWare USB2 DRD Core Support
                Driver Mode (Device Mode Only)  --->
```

第二步：编译内核 make uImage 并烧录到开发板上；

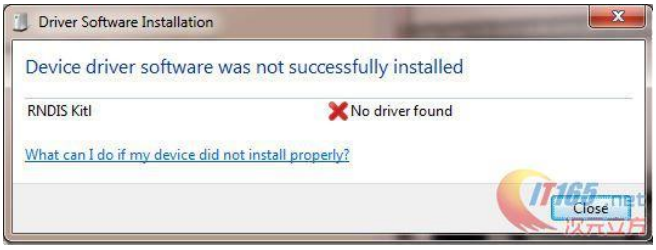
第三步：等待进入系统以后

```
$ echo connect > /sys/devices/platform/jz-dwc2/dwc2/udc/dwc2/soft_connect
$ ifconfig usb0 193.169.4.xx up
$ route add default gw 193.169.4.1
```

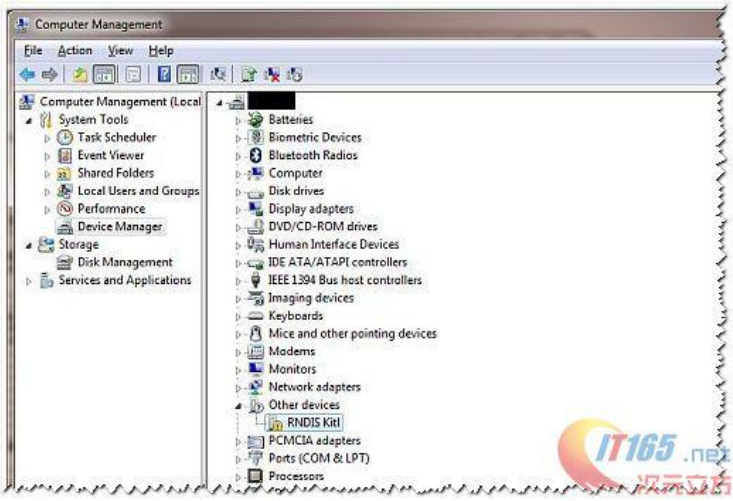
第四步：PC 端配置

1) RNDIS（Remote Network Driver Interface Specification）也叫远端网络驱动接口协议，设备通过 USB 方式同主机连接，模拟网络连接以便用于下载和调试工作。RNDIS 驱动是 Windows7 的一部分，遗憾的是如果默认安装（插上符合 RNDIS 的设备时）一般均会安装失败（以后可能会修正此问题）。可通过如下方法重新安装 RNDIS 驱动。

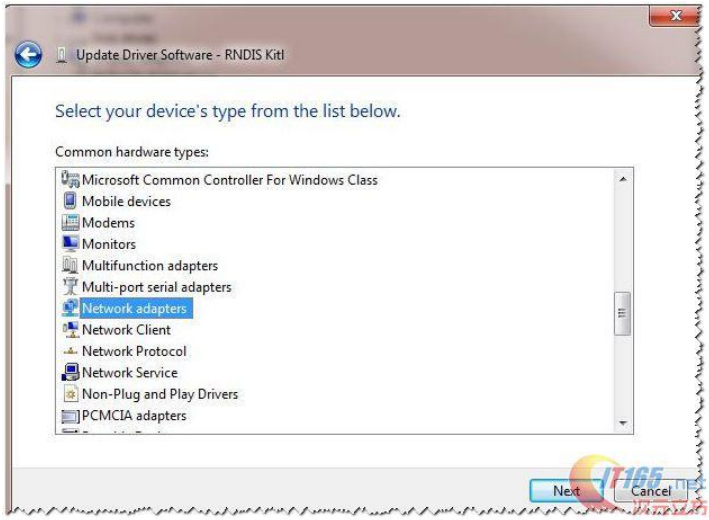
2) 设备同计算机连接时，操作系统会自动搜索并安装 RNDIS 驱动，不过，片刻之后您会发现安装失败。



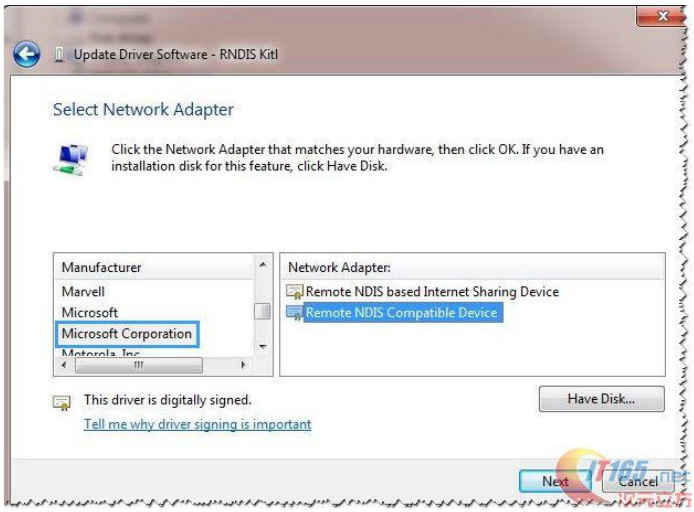
3) 右键点击桌面“计算机”图标，选择“管理”——“设备管理”，可以看到“RNDIS Kitl”设备，并且处于驱动未安装状态。



4) 右键点击该设备，选择更新驱动软件，在如何搜索设备软件提示窗口中，选择“浏览我的计算机”。选择从设备列表中选择“网络适配器”。

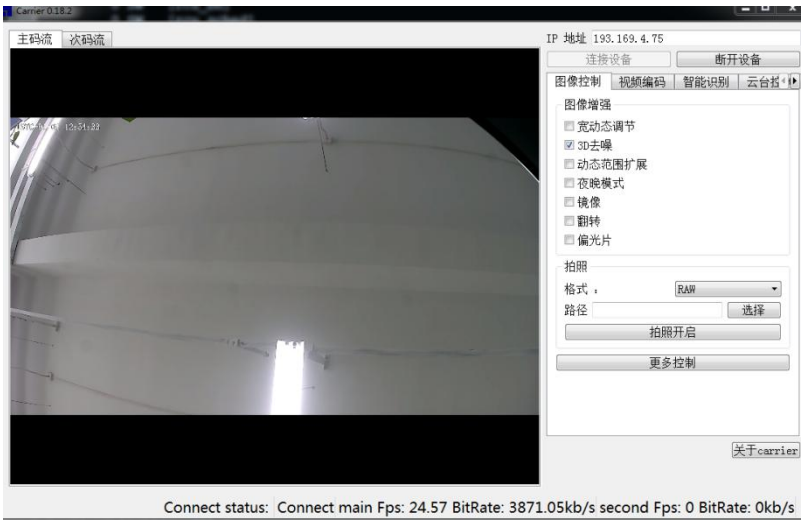


5) 在网络适配器窗口的制造商列表中选择微软公司（Microsoft Corporation），右侧列表中选择远端 NDIS 兼容设备。



6) 点击下一步并等待安装结束，RNDIS Kitl 设备将会安装成功。

7) 然后可以执行 carrier tool 工具查看图像。

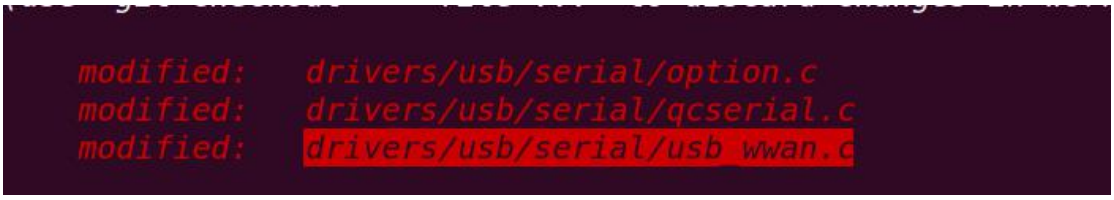


6.14 4G

6.14.1 PPP

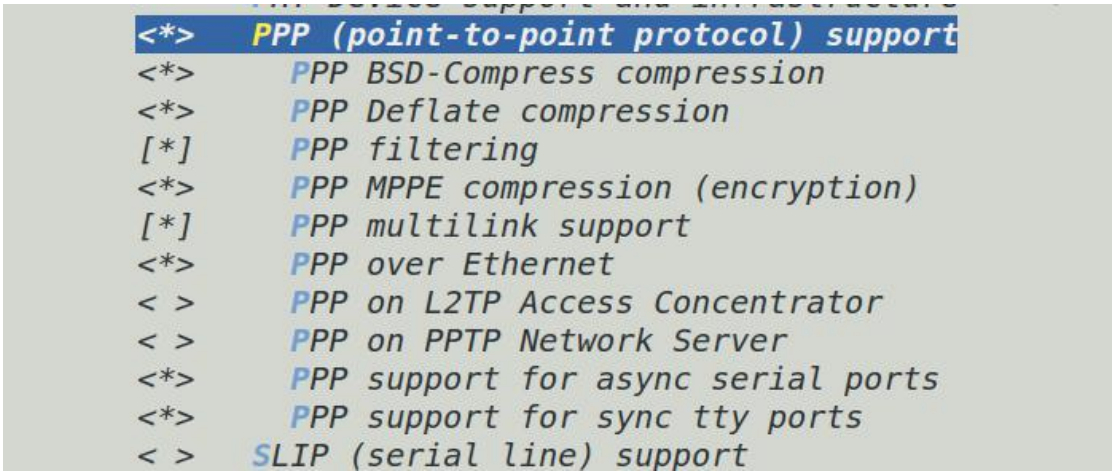
第一步：内核中需要修改的代码

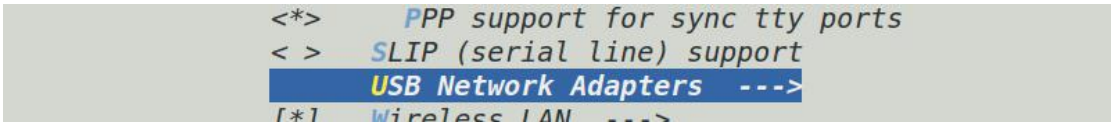
请使用本文件夹下面的文件替换内核中文件

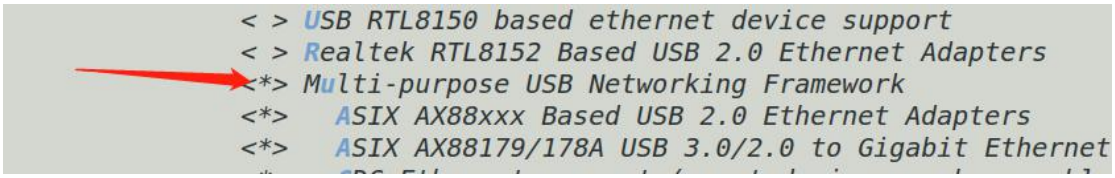


第二步：内核配置（编译烧录）

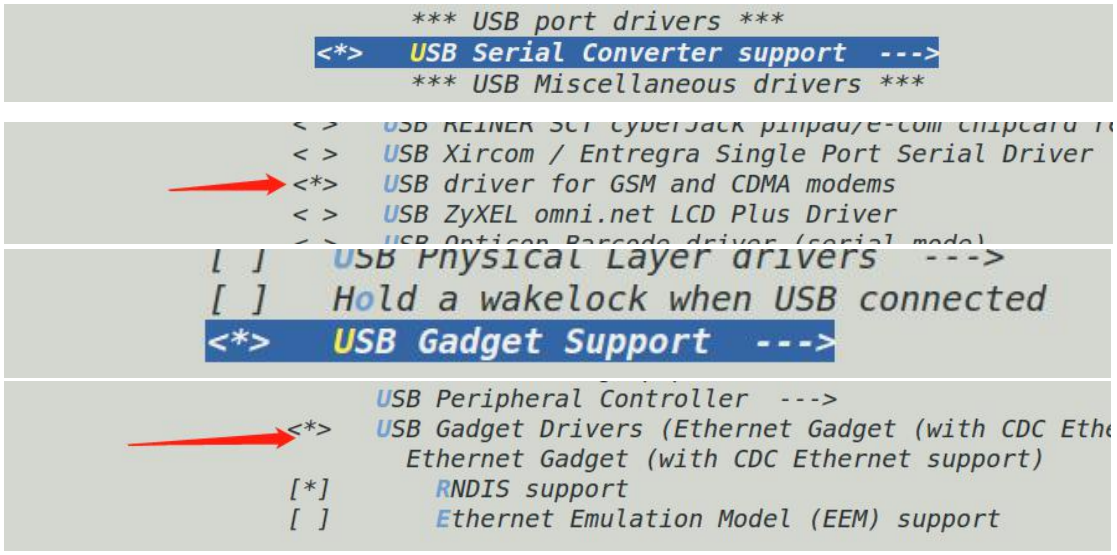
A. network 部分修改







B. usb 部分修改



第三步：文件系统的修改

A. 请在文件系统的 /etc 目录下通过创建 ppp 软连接；可参考如下操作

```
cd /etc
ln -s ../system/etc/ppp ppp
```

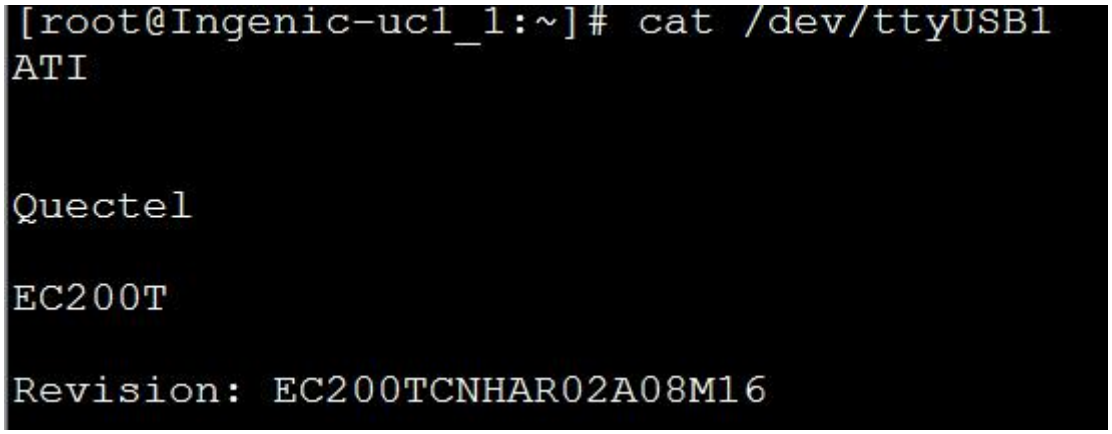
B. 制作 squashfs 文件系统。

linux-ppp-scripts 文件夹里面的 lib 库拷贝到文件系统中， sbin 下的拷贝到系统的 bin 下面;其它脚本参考 readme 使用方法一,拷贝到 /etc/ppp/peers 文件夹下面;ip-up 需要拷贝到 /etc/ppp 目录下！

第四步：系统调试

A. 优先查看是否生成了 /dev/ttyUSB*

B. 执行 `echo -e "ATI\r\n" > /dev/ttyUSB1; cat /dev/ttyUSB1`



出现上图所示；表明可以和 4G 模块通信了！

C. 执行命令 pppd call quectel-ppp &

```
Script chat -s -v -f /etc/ppp/peers/quectel-chat-connect finished (pid 299), status = 0x0
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyUSB2
sent [LCP ConfReq id=0x1 <asyncmap 0x0> <magic 0x97e58dc8> <pcomp> <accomp>]
rcvd [LCP ConfReq id=0x1 <asyncmap 0x0> <auth pap> <magic 0x33bafefe> <pcomp> <accomp>]
sent [LCP ConfAck id=0x1 <asyncmap 0x0> <auth pap> <magic 0x33bafefe> <pcomp> <accomp>]
rcvd [LCP ConfAck id=0x1 <asyncmap 0x0> <magic 0x97e58dc8> <pcomp> <accomp>]
sent [PAP AuthReq id=0x1 user="test" password=<hidden>]
rcvd [PAP AuthAck id=0x1 "" 00]
PAP authentication succeeded
sent [IPCP ConfReq id=0x1 <addr 0.0.0.0> <ms-dns1 0.0.0.0> <ms-dns2 0.0.0.0>]
rcvd [IPCP ConfReq id=0x2]
sent [IPCP ConfNak id=0x2 <addr 0.0.0.0>]
rcvd [IPCP ConfNak id=0x1 <addr 100.185.143.235> <ms-dns1 202.102.213.68> <ms-dns2 61.132.163.68>]
sent [IPCP ConfReq id=0x2 <addr 100.185.143.235> <ms-dns1 202.102.213.68> <ms-dns2 61.132.163.68>]
rcvd [IPCP ConfReq id=0x3]
sent [IPCP ConfAck id=0x3]
rcvd [IPCP ConfAck id=0x2 <addr 100.185.143.235> <ms-dns1 202.102.213.68> <ms-dns2 61.132.163.68>]
Could not determine remote IP address: defaulting to 10.64.64.64
local  IP address 100.185.143.235
remote IP address 10.64.64.64
primary   DNS address 202.102.213.68
secondary DNS address 61.132.163.68

[root@Ingenic-uc1_1:~]# ping www.baidu.com
ping: bad address 'www.baidu.com'
[root@Ingenic-uc1_1:~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         0.0.0.0         0.0.0.0         U     0      0        0 ppp0
10.64.64.64     0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
[root@Ingenic-uc1_1:~]#
```

出现上图打印说明配网成功！

D. 检测 /etc/resolv.conf 是否有配置 DNS，如果有就可以进行下一步

```
[root@Ingenic-uc1_1:~]# cat /etc/resolv.conf
nameserver 61.132.163.68
nameserver 202.102.213.68
```

E. ping www.baidu.com 测试

```
[root@Ingenic-uc1_1:~]# ping www.baidu.com
PING www.baidu.com (14.215.177.38): 56 data bytes
64 bytes from 14.215.177.38: seq=0 ttl=54 time=38.556 ms
64 bytes from 14.215.177.38: seq=1 ttl=54 time=42.371 ms
64 bytes from 14.215.177.38: seq=2 ttl=54 time=37.308 ms
64 bytes from 14.215.177.38: seq=3 ttl=54 time=37.153 ms
```

6.14.2 GobNet

第一步：在配置 gobnet 时需要先在内核中修改以下代码：

```
A. /drivers/net/usb/Makefile (添加下面两行)
obj-y += GobiNet.o
GobiNet-objs := GobiUSBNet.o QMIDevice.o QMI.o
B. /drivers/net/usb/qmi_wwan.c  (注释下面一行代码)
// {QMI_GOBI_DEVICE(0x05c6, 0x9215)},      /* Acer Gobi 2000 Modem device (VP413) */
C. /drivers/usb/serial/qcserial.c  (注释下面一行代码)
// {USB_DEVICE(0x05c6, 0x9215)},   /* Acer Gobi 2000 Modem device (VP413) */
```

第二步：内核配置（编译烧录）

```
[*] Device Drivers →
```

```
    -*- Network device support →
        USB Network Adapters →
            {*} Multi-purpose USB Networking Framework
```

第三步：烧录完内核后执行 quectel-CM &即可

6.14.3 WWAN

第一步：在配置 gobnet 时需要先在内核中修改以下代码：参考附录 7.3 代码

第二步：内核配置（编译烧录）

```
 [*] Device Drivers →
    -*- Network device support →
        USB Network Adapters →
            {*} Multi-purpose USB Networking Framework
                <*> QMI WWAN driver for Qualcomm MSM based 3G and LTE modems
```

第三步：烧录完内核后执行 quectel-CM &即可。

7 附录

7.1 制作启动卡

对于第一次烧录 Nor，或者没有网络的情况来说，需要从 SD 卡启动，并将 uboot 烧录到 Nor 中。之后的烧录就可以通过 tftpboot 进行。

1）格式化 sd 卡，并将卡启动 uboot 烧录到 SD 卡

此步骤只需执行一次，若不更新卡起 uboot 则不需要再次执行此步骤。经过此步骤处理的 SD 卡可当作正常卡使用。

将 SD 卡插入电脑

A. 卸载 sd 卡

```
umount /media/user/*
```

B. 将 SD 卡重新分区

```
fdisk /dev/sdb
> o  --创建一个新的分区表
> n  --添加一个分区 n
--此时会提示：
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
Using default response p
--这里回车默认是 p，创建主分区
分区号 (1-4，默认为 1)：
将使用默认值 1
--这里回车默认创建 1 号分区
起始 sector (2048-15130623，默认为 2048)：
将使用默认值 2048
Last sector, +扇区 or +size{K,M,G} (2048-15130623，默认为 15130623)：
将使用默认值 15130623
--2048 号扇区在 1MB 的位置，给 uboot 空出了空间
最后一步，输入 w，向 SD 卡中写入分区表
The partition table has been altered!
Calling ioctl() to re-read partition table.
```

```
Syncing disks.
```

到这里分区表已创建完成

D．执行 sync，拔卡，重新插卡，PC 将重新识别分区

E．格式化 sd 卡为通用的 VFat 文件系统：`mkfs.vfat /dev/sdb1`

F．编译 u-boot。

```
make isvp_t31_msc0   //编译对应型号卡启的 uboot
```

将生成的 u-boot-with-spl.bin 文件烧录进 sd 卡的 17KB 偏移处：

```
dd if=u-boot-with-spl.bin of=/dev/sdb bs=1024 seek=17
```

注意！！！：确定插入你的电脑上 TF 卡对应的设备节点文件是不是 sdb(有可能是 sdc 或者 sdd)；如果对应错了可能会破坏电脑的硬盘文件系统。

到此，可以用来烧录的 SD 卡制作完成。

插入 SD 卡，复制 u-boot-with-spl.bin，uImage，rootfs_glibc.jffs2 等粘贴到 sd 卡的正常文件系统中。

3）使用 sd 卡启动开发板，开机前几秒迅速按下回车键，进入 uboot 命令行

A．首先，将需要烧录的文件读到内存中，以 u-boot 为例：

B．把内存先全部清为全 f

`mw.b 0x80600000 0xff 0x1000000` --第三个参数是大小，这里清了 16MB

C．ls SD 卡中的文件

```
isvp# fatls mmc 0
   222944   u-boot-with-spl.bin
   2368879   uimage
5 file(s), 0 dir(s)
```

D．load 文件到内存

```
fatload mmc 0 0x80600000 u-boot-with-spl.bin
```

E.然后，将内存中的数据写入 Nor

```
sf probe
sf erase 0x0 0x40000  --擦除 uboot 分区大小 256K
sf write 0x80600000 0x0 0x40000  --将 uboot 写入 Nor 0x0~0x40000 位置处
```

7.2 wpa_supplicant 使用方法

wpa_supplicant 及 wpa_cli 使用方法。

7.2.1 概述

wpa_supplicant 是一个连接、配置 WIFI 的工具。它主要包含两个程序：wpa_supplicant 与 wpa_cli。二者的关系就是 server 与 client 的关系。通常情况下，可以通过 wpa_cli 来进行 WIFI 的配置与连接，如果有特殊的需要，可以编写应用程序直接调用 wpa_supplicant 的接口直接开发。

本文主要讲述如何通过 wpa_cli 进行 WIFI 的配置与连接。

7.2.2 使用方法

1）启动 wpa_supplicant 应用

```
$ wpa_supplicant -D nl80211 -i wlan0 -c /etc/wpa_supplicant.conf -B
```

注意：/etc/wpa_supplicant.conf 文件里，添加下面代码。

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
```

2）启动 wpa_cli 应用

```
$ wpa_cli -i wlan0 scan                搜索附近 wifi 网络
$ wpa_cli -i wlan0 scan_result         打印搜索 wifi 网络结果
$ wpa_cli -i wlan0 add_network         添加一个网络连接
如果要连接加密方式是：[WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][ESS] (wpa 加密)
wifi 名称是：wifi_name
wifi 密码是：wifi_psk
$ wpa_cli -i wlan0 set_network 0 ssid '"wifi_name"'
$ wpa_cli -i wlan0 set_network 0 psk '"wifi_psk"'
$ wpa_cli -i wlan0 enable_network 0

如果要连接加密方式是：[WEP][ESS] (wep 加密)
wifi 名称是：wifi_name
wifi 密码是：wifi_psk
$ wpa_cli -i wlan0 set_network 0 ssid '"wpa_name"'
$ wpa_cli -i wlan0 set_network 0 key_mgmt NONE
$ wpa_cli -i wlan0 set_network 0 wep_key0 '"wap_psk"'
$ wpa_cli -i wlan0 enable_network 0

如果要连接加密方式是：[ESS] (无加密)
wifi 名称是：wifi_name
$ wpa_cli -i wlan0 set_network 0 ssid '"wifi_name"'
$ wpa_cli -i wlan0 set_network 0 key_mgmt NONE
$ wpa_cli -i wlan0 enable_network 0
```

3） 分配 ip,netmask,gateway,dns

```
$ udhcpc -i wlan0
```

执行完毕，即可以连接网络。

4） 保存连接

```
$ wpa_cli -i wlan0 save_config
```

5）断开连接

```
$ wpa_cli -i wlan0 disable_network 0
```

6）连接已有的连接

```
$ wpa_cli -i wlan0 list_network              列举所有保存的连接
```

```
$ wpa_cli -i wlan0 select_network 0          连接第 1 个保存的连接
$ wpa_cli -i wlan0 enable_network 0          使能第 1 个保存的连接
```

7）断开 wifi

```
$ ifconfig wlan0 down
$ killall udhcpc
$ killall wpa_supplicant
```

7.2.3 wpa_wifi_tool 使用方法

wpa_wifi_tool 是基于 wpa_supplicant 及 wpa_cli 的一个用于快速设置 wifi 的工具，方便调试时连接 wifi 使用。使用者无需运行步骤 2 中复杂的命令，即可实现 wifi 设置和连接等功能。

注：此工具仅作为调试工具使用，实际 wifi 功能开发推荐使用 wpa_cli 或者直接调用 wpa_supplicant 实现。

使用方法：

A. 运行 wpa_wifi_tool

B. 输入 help 进行命令查看

C. s 进行 SSID 扫描

D. c[n]进行 wifi 连接，连接时若为新的 SSID 则需输入密码，若为已保存的 SSID 则可以使用保存过的密码或者重新输入密码

E. e 退出工具

其他使用方法请参考 help

7.3 4G 代码参考

A drivers/net/usb/qmi_wwan.c 添加如下代码

```
--- a/drivers/net/usb/qmi_wwan.c
+++ b/drivers/net/usb/qmi_wwan.c
@@ -20,6 +20,72 @@
 #include <linux/usb/usbnet.h>
 #include <linux/usb/cdc-wdm.h>

+#if 1 //Added by Quectel
+#include <linux/etherdevice.h>
+struct sk_buff *qmi_wwan_tx_fixup(struct usbnet *dev, struct sk_buff *skb,
gfp_t flags)
+{
+	if (dev->udev->descriptor.idVendor != cpu_to_le16(0x2C7C))
+		return skb;
+	//Skip Ethernet header from message
+	if (dev->net->hard_header_len == 0)
```

```
+               return skb;
+       else
+               skb_reset_mac_header(skb);
+       if (skb_pull(skb, ETH_HLEN)) {
+               return skb;
+       } else {
+               dev_err(&dev->intf->dev, "Packet Dropped ");
+       }
+       // Filter the packet out, release it
+       dev_kfree_skb_any(skb);
+       return NULL;
+}
+
+#include <linux/version.h>
+#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
+static int qmi_wwan_rx_fixup(struct usbnet *dev, struct sk_buff *skb)
+{
+       __be16 proto;
+       if (dev->udev->descriptor.idVendor != cpu_to_le16(0x2C7C))
+               return 1;
+       /* This check is no longer done by usbnet */
+       if (skb->len < dev->net->hard_header_len)
+               return 0;
+       switch (skb->data[0] & 0xf0) {
+               case 0x40:
+                       proto = htons(ETH_P_IP);
+                       break;
+               case 0x60:
+                       proto = htons(ETH_P_IPV6);
+                       break;
+               case 0x00:
+                       if (is_multicast_ether_addr(skb->data))
+                               return 1;
+                       /* possibly bogus destination - rewrite just in case
*/
+                       skb_reset_mac_header(skb);
+                       goto fix_dest;
+               default:
+                       /* pass along other packets without modifications */
+                       return 1;
+       }
+       if (skb_headroom(skb) < ETH_HLEN)
```

```
+             return 0;
+      skb_push(skb, ETH_HLEN);
+      skb_reset_mac_header(skb);
+      eth_hdr(skb)->h_proto = proto;
+      memset(eth_hdr(skb)->h_source, 0, ETH_ALEN);
+fix_dest:
+      memcpy(eth_hdr(skb)->h_dest, dev->net->dev_addr, ETH_ALEN);
+      return 1;
+}
+/* very simplistic detection of IPv4 or IPv6 headers */
+static bool possibly_iphdr(const char *data)
+{
+      return (data[0] & 0xd0) == 0x40;
+}
+#endif
+#endif
+
 /* This driver supports wwan (3G/LTE/?) devices using a vendor
  * specific management protocol called Qualcomm MSM Interface (QMI) -
  * in addition to the more common AT commands over serial interface
@@ -332,6 +398,33 @@ next_desc:
              dev->net->dev_addr[0] &= 0xbf;  /* clear "IP" bit */
       }
       dev->net->netdev_ops = &qmi_wwan_netdev_ops;
+
+#if 1 //Added by Quectel
+      if (dev->udev->descriptor.idVendor == cpu_to_le16(0x2C7C)) {
+             if (intf->cur_altsetting->desc.bInterfaceClass != 0xff) {
+                    dev_dbg(&intf->dev, "Quectel module not qmi_wwan mode! please check 'at+qcfg=\"usbnet\"'\n");
+                    return -ENODEV;
+             }
+             dev_dbg(&intf->dev, "Quectel EC25&EC21&EG91&EG95&EG06&EP06&EM06&EG12&EP12&EM12&EG16&EG18&BG96&AG35 work on RawIP mode\n");
+             dev->net->flags |= IFF_NOARP;
+#if (LINUX_VERSION_CODE < KERNEL_VERSION( 3,9,1 ))
+             /* make MAC addr easily distinguishable from an IP header */
+             if (possibly_iphdr(dev->net->dev_addr)) {
+                    dev->net->dev_addr[0] |= 0x02; /* set local assignment bit */
```

```
    +                       dev->net->dev_addr[0] &= 0xbf; /* clear "IP" bit */
    +               }
    +#endif
    +               usb_control_msg(
    +                               interface_to_usbdev(intf),
    +                               usb_sndctrlpipe(interface_to_usbdev(intf),
0),
    +                               0x22, //USB_CDC_REQ_SET_CONTROL_LINE_STATE
    +                               0x21, //USB_DIR_OUT | USB_TYPE_CLASS |
USB_RECIP_INTERFACE
    +                               1, //active CDC DTR
    +
intf->cur_altsetting->desc.bInterfaceNumber,
    +                               NULL, 0, 100);
    +       }
    +#endif
    +
     err:
            return status;
     }
    @@ -416,6 +509,10 @@ static const struct driver_info    qmi_wwan_info = {
            .unbind         = qmi_wwan_unbind,
            .manage_power   = qmi_wwan_manage_power,
            .rx_fixup       = qmi_wwan_rx_fixup,
    +#if 1 //Added by Quectel
    +       .tx_fixup = qmi_wwan_tx_fixup,
    +       .rx_fixup = qmi_wwan_rx_fixup,
    +#endif
     };

     #define HUAWEI_VENDOR_ID       0x12D1
    @@ -434,6 +531,31 @@ static const struct driver_info    qmi_wwan_info = {
            QMI_FIXED_INTF(vend, prod, 0)

     static const struct usb_device_id products[] = {
    +#if 1 //Added by Quectel
    +#ifndef QMI_FIXED_INTF
    +       /* map QMI/wwan function by a fixed interface number */
    +#define QMI_FIXED_INTF(vend, prod, num) \
    +       .match_flags = USB_DEVICE_ID_MATCH_DEVICE |
    +       USB_DEVICE_ID_MATCH_INT_INFO, \
    +               .idVendor = vend, \
```

```
    +               .idProduct = prod, \
    +               .bInterfaceClass = 0xff, \
    +               .bInterfaceSubClass = 0xff, \
    +               .bInterfaceProtocol = 0xff, \
    +               .driver_info = (unsigned long)&qmi_wwan_force_int##num,
    +#endif
    +       { QMI_FIXED_INTF(0x05C6, 0x9003, 4) }, /* Quectel UC20 */
    +       { QMI_FIXED_INTF(0x2C7C, 0x0125, 4) }, /* Quectel EC25 */
    +       { QMI_FIXED_INTF(0x2C7C, 0x0121, 4) }, /* Quectel EC21 */
    +       { QMI_FIXED_INTF(0x05C6, 0x9215, 4) }, /* Quectel EC20 */
    +       { QMI_FIXED_INTF(0x2C7C, 0x0191, 4) }, /* Quectel EG91 */
    +       { QMI_FIXED_INTF(0x2C7C, 0x0195, 4) }, /* Quectel EG95 */
    +       { QMI_FIXED_INTF(0x2C7C, 0x0306, 4) }, /* Quectel EG06/EP06/EM06
*/
    +       { QMI_FIXED_INTF(0x2C7C, 0x0512, 4) }, /* Quectel
EG12/EP12/EM12/EG16/EG18 */
    +       { QMI_FIXED_INTF(0x2C7C, 0x0296, 4) }, /* Quectel BG96 */
    +       { QMI_FIXED_INTF(0x2C7C, 0x0435, 4) }, /* Quectel AG35 */
    +#endif
    +
            /* 1. CDC ECM like devices match on the control interface */
            {       /* Huawei E392, E398 and possibly others sharing both device
id and more... */
                    USB_VENDOR_AND_INTERFACE_INFO(HUAWEI_VENDOR_ID,
USB_CLASS_VENDOR_SPEC, 1, 9),
    @@ -614,7 +736,7 @@ static const struct usb_device_id products[] = {
            {QMI_GOBI_DEVICE(0x05c6, 0x9225)},      /* Sony Gobi 2000 Modem
device (N0279, VU730) */
            {QMI_GOBI_DEVICE(0x05c6, 0x9245)},      /* Samsung Gobi 2000 Modem
device (VL176) */
            {QMI_GOBI_DEVICE(0x03f0, 0x251d)},      /* HP Gobi 2000 Modem
device (VP412) */
    -       {QMI_GOBI_DEVICE(0x05c6, 0x9215)},      /* Acer Gobi 2000 Modem
device (VP413) */
    +//     {QMI_GOBI_DEVICE(0x05c6, 0x9215)},      /* Acer Gobi 2000 Modem
device (VP413) */
            {QMI_GOBI_DEVICE(0x05c6, 0x9265)},      /* Asus Gobi 2000 Modem
device (VR305) */
            {QMI_GOBI_DEVICE(0x05c6, 0x9235)},      /* Top Global Gobi 2000
Modem device (VR306) */
            {QMI_GOBI_DEVICE(0x05c6, 0x9275)},      /* iRex Technologies Gobi
2000 Modem device (VR307) */
```

B drivers/net/usb/qmi_wwan.c

```
 --- a/drivers/usb/serial/option.c
+++ b/drivers/usb/serial/option.c
@@ -530,6 +530,19 @@ static const struct option_blacklist_info
telit_le920_blacklist = {
 };

 static const struct usb_device_id option_ids[] = {
+#if 1 //Added by Quectel
+       { USB_DEVICE(0x05C6, 0x9090) }, /* Quectel UC15 */
+       { USB_DEVICE(0x05C6, 0x9003) }, /* Quectel UC20 */
+       { USB_DEVICE(0x2C7C, 0x0125) }, /* Quectel EC25 */
+       { USB_DEVICE(0x2C7C, 0x0121) }, /* Quectel EC21 */
+       { USB_DEVICE(0x05C6, 0x9215) }, /* Quectel EC20 */
+       { USB_DEVICE(0x2C7C, 0x0191) }, /* Quectel EG91 */
+       { USB_DEVICE(0x2C7C, 0x0195) }, /* Quectel EG95 */
+       { USB_DEVICE(0x2C7C, 0x0306) }, /* Quectel EG06/EP06/EM06 */
+       { USB_DEVICE(0x2C7C, 0x0512) }, /* Quectel EG12/EP12/EM12/EG16/EG18
*/
+       { USB_DEVICE(0x2C7C, 0x0296) }, /* Quectel BG96 */
+       { USB_DEVICE(0x2C7C, 0x0435) }, /* Quectel AG35 */
+#endif
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_COLT) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA) },
        { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_LIGHT) },
@@ -1245,7 +1258,7 @@ static const struct usb_device_id option_ids[] = {
        { USB_DEVICE(CINTERION_VENDOR_ID, CINTERION_PRODUCT_AHXX) },
        { USB_DEVICE(CINTERION_VENDOR_ID, CINTERION_PRODUCT_PLXX),
              .driver_info = (kernel_ulong_t)&net_intf4_blacklist },
-       { USB_DEVICE(CINTERION_VENDOR_ID, CINTERION_PRODUCT_HC28_MDM) },
+       { USB_DEVICE(CINTERION_VENDOR_ID, CINTERION_PRODUCT_HC28_MDM) },
        { USB_DEVICE(CINTERION_VENDOR_ID,
CINTERION_PRODUCT_HC28_MDMNET) },
        { USB_DEVICE(SIEMENS_VENDOR_ID, CINTERION_PRODUCT_HC25_MDM) },
        { USB_DEVICE(SIEMENS_VENDOR_ID, CINTERION_PRODUCT_HC25_MDMNET) },
@@ -1377,6 +1390,9 @@ static struct usb_serial_driver option_1port_device
= {
 #ifdef CONFIG_PM
        .suspend           = usb_wwan_suspend,
        .resume            = usb_wwan_resume,
+#if 1 //Added by Quectel
```

```
    +       .reset_resume = usb_wwan_resume,
    +#endif
     #endif
     };

    @@ -1444,6 +1460,22 @@ static int option_probe(struct usb_serial *serial,
               iface_desc->bInterfaceClass != USB_CLASS_CDC_DATA)
                   return -ENODEV;

    +#if 1 //Added by Quectel
    +       //Quectel UC20's interface 4 can be used as USB network device
    +       if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6)
    +                       && serial->dev->descriptor.idProduct ==
cpu_to_le16(0x9003)
    +                       &&
serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
    +               return -ENODEV;
    +       //Quectel EC20's interface 4 can be used as USB network device
    +       if (serial->dev->descriptor.idVendor == cpu_to_le16(0x05C6)
    +                       && serial->dev->descriptor.idProduct ==
cpu_to_le16(0x9215)
    +                       &&
serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
    +               return -ENODEV;
    +       //Quectel EC25&EC21&EG91&EG95&EG06&EP06&EM06&BG96/AG35's
interface 4 can be used as USB network device
    +       if (serial->dev->descriptor.idVendor == cpu_to_le16(0x2C7C)
    +                       &&
serial->interface->cur_altsetting->desc.bInterfaceNumber >= 4)
    +               return -ENODEV;
    +#endif
            /* Store device id so we can use it during attach. */
            usb_set_serial_data(serial, (void *)id);
```

C drivers/usb/serial/qcserial.c

```
    --- a/drivers/usb/serial/qcserial.c
    +++ b/drivers/usb/serial/qcserial.c
    @@ -78,7 +78,7 @@ static const struct usb_device_id id_table[] = {
            {USB_DEVICE(0x03f0, 0x241d)},   /* HP Gobi 2000 QDL device (VP412)
*/
            {USB_DEVICE(0x03f0, 0x251d)},   /* HP Gobi 2000 Modem device (VP412)
*/
```

```
        {USB_DEVICE(0x05c6, 0x9214)},   /* Acer Gobi 2000 QDL device (VP413)
*/
    -       {USB_DEVICE(0x05c6, 0x9215)},   /* Acer Gobi 2000 Modem device
(VP413) */
    +//     {USB_DEVICE(0x05c6, 0x9215)},   /* Acer Gobi 2000 Modem device
(VP413) */
        {USB_DEVICE(0x05c6, 0x9264)},   /* Asus Gobi 2000 QDL device (VR305)
*/
        {USB_DEVICE(0x05c6, 0x9265)},   /* Asus Gobi 2000 Modem device
(VR305) */
        {USB_DEVICE(0x05c6, 0x9234)},   /* Top Global Gobi 2000 QDL device
(VR306) */
    diff --git a/drivers/usb/serial/usb_wwan.c
b/drivers/usb/serial/usb_wwan.c
    index db0cf53..c0d05a4 100644
    --- a/drivers/usb/serial/usb_wwan.c
    +++ b/drivers/usb/serial/usb_wwan.c
    @@ -459,6 +459,20 @@ static struct urb *usb_wwan_setup_urb(struct
usb_serial_port *port,
                            usb_sndbulkpipe(serial->dev, endpoint) | dir,
                            buf, len, callback, ctx);

    +#if 1 //Added by Quectel for zero packet
    +       if (dir == USB_DIR_OUT) {
    +               struct usb_device_descriptor *desc =
&serial->dev->descriptor;
    +               if (desc->idVendor == cpu_to_le16(0x05C6) &&
desc->idProduct == cpu_to_le16(0x9090))
    +                       urb->transfer_flags |= URB_ZERO_PACKET;
    +               if (desc->idVendor == cpu_to_le16(0x05C6) &&
desc->idProduct == cpu_to_le16(0x9003))
    +                       urb->transfer_flags |= URB_ZERO_PACKET;
    +               if (desc->idVendor == cpu_to_le16(0x05C6) &&
desc->idProduct == cpu_to_le16(0x9215))
    +                       urb->transfer_flags |= URB_ZERO_PACKET;
    +               if (desc->idVendor == cpu_to_le16(0x2C7C))
    +                       urb->transfer_flags |= URB_ZERO_PACKET;
    +       }
    +#endif
    +
        return urb;
     }
```

7.4 版本说明

ISVP_Devkit 版本升级说明

ISVP 版本升级（主要以软件资源及 SDK 升级为主）是一项持续的工作，新的版本会提供新的功能、修复问题、以及系统优化。

ISVP 软件系统主要包含：uboot，kernel，drivers，lib，sensor bin，rootfs，toolchain 几个部分，对于每次 Devkit 更新，这几个部分间可能存在一定的依赖关系，需要同步升级，否则可能会出现错误。

版本升级有以下注意事项：

A. 开发人员需要认真阅读 ChangeLog，了解有哪些更新。每次的 ChangeLog 会注释“务必更新”和“推荐更新”的资源。

B. 一般情况下建议 drivers(ISP, Sensor)、sensor bin、与 lib 同时更新。

C. kernel 更新较少，但是如果需要更新 kernel，需要完全重新编译所有 ko，其中 ISP driver 和 Sensor driver 的编译顺序是先编译前者后编译后者。

D. ISP driver 与 Sensor driver 及 Sensor bin 文件三者有匹配关系，不匹配可能造成图像异常。

E. API 更新需要完全替换 lib 与头文件，头文件与 lib 不匹配可能造成程序崩溃。

新的版本也可能会带来新的问题，ISVP 的开发团队会尽力保证新版本的正确性。若开发者发现了 Bug，请及时向君正的技术支持人员反馈，描述现象与复现情况等细节，ISVP 的开发团队会第一时间进行处理。

调试过程中需要问题，请与技术支持人员进行沟通反馈。正确的提供 Log 以及记录问题，可以有效提高解决问题的效率。主要有以下几点：

A. 正确的描述现象，复现方法，复现概率

B. 提供 Log 以及截图

C. 提供 SDK 版本信息（或提供完整的 logcat）

D. 找到复现规律

7.5 调试说明

7.5.1 ISP 如何抓 RAW 图

当我们需要获取 RAW 原始图像数据时，我们可以在开发板上直接输出命令抓取图像数据。

命令使用帮助：

```
$ echo help > proc/jz/isp/isp-w02
```

抓取图像：

```
$ echo snapraw [savenum] > /proc/jz/isp/isp-w02    //savenum 抓取的图像数
```

保存图像：

```
$ echo snapraw [savenum] > /proc/jz/isp/isp-w02    //savenum 保存的图像数
```

最终保存的图像数据存放在 /tmp 目录下。

7.5.2 如何抓取 Log

出现问题时，需要抓取 kernel 以及 lib 的 log。

kernel log 抓取命令：

```
$ dmesg > /tmp/time.dmesg。
```

lib log 抓取方法：

```
$ logcat -c time > /tmp/time.log
```

命令大概运行 3 秒后 ctrl+c 终止程序（logcat 命令在 sdk tools 目录下）。

7.5.3 读写 eth phy 寄存器

PHY 是 IEEE802.3 中定义的一个标准模块，可以对 PHY 的行为、状态进行管理和控制，而具体管理和控制动作是通过读写 PHY 内部的寄存器实现的。

Uboot 环境下：

读格式：ethphy read addr

```
isvp_t31# ethphy read 0
```

写格式：ethphy write addr data

```
isvp_t31# ethphy write 0 0
```

Kernel 环境下：

读格式：reg:phy 地址-phy 寄存器地址(phy 地址默认是 0，phy 寄存器地址根据标准手册查看)

```
$ echo r reg:0-0 > /proc/jz/mdio/cmd
```

写格式：reg:phy 地址-phy 寄存器地址-写入数据

```
$ echo w reg:0-0-0 > /proc/jz/mdio/cmd
```

7.5.4 RMEM 查看与设置

1) 内存的使用说明

在 uboot 的 cmdline 里面有下面内存配置

```
mem=42012K@0x0 ispmem=8100K@0x2907000 rmem=15424K@0x30F0000
mem 是分配给 linux 系统使用的内存。
rmem 是 SDK 保留使用的内存，使用该内存的地方包括，ISP dmaout 的轮转 buf、编码模块的输入输出 buf，OSD 模块的叠加图片。
```

2) rmem 内存大小如何确定

```
A. rmem 的内存可以使用 SDK 中的 rmem 计算器 excel 表格来计算得到。
B. 先把 rmem 设置大一下，先让系统跑起来，然后执行 impdbg --sys;logcat 可以看到 rmem 实际使用的内存和剩余内存，然后把剩余内存从 rmem 拿出来给 mem 使用。
```

3) rmem 内存的优化

OSD 内存占用默认是 1M，可以根据实际产品功能调小，例如 512K。可以通过下面接口设置 OSD 的 rmem 的内存占用。OSD 占用的内存大小是叠加最大图片的 width*height*4。

内存优化调用(默认是 1M)

```
extern "C" {
```

```
//设置 OSD 模块使用大小
extern int IMP_OSD_SetPoolSize(int size);
}
初始化 SDK 之前调用
IMP_OSD_SetPoolSize(512*1024);
```

7.5.5 如何保留&恢复现场

对于 Nor Flash，可以把 Flash 内容保存下来。方法如下：

```
$ dd if=/dev/mtd0 of=/tmp/mtd0
$ dd if=/dev/mtd1 of=/tmp/mtd1
$ dd if=/dev/mtd2 of=/tmp/mtd2
$ dd if=/dev/mtd3 of=/tmp/mtd3
$ cd tmp
$ cat mtd0 mtd1 mtd2 mtd3 > flash.bin
$ rm mtd0 mtd1 mtd2 mtd3
把 flash.bin 取出即可
或者：
$ cat /dev/mtd0 /dev/mtd1 /dev/mtd2 /dev/mtd3 > flash.bin
```

在需要恢复现场时，在 u-boot 中把 Nor Flash 完全擦除，再把 flash.bin 完全写入即可。

7.5.6 Coredump 使用方法

7.5.6.1 coredump 介绍

当程序运行的过程中异常终止或者崩溃，操作系统会将程序当时的内存状态记录下来，保存在一个文件中，这种行为就叫做 core dump。可以认为 core dump 是“内存快照”，但实际上，除了内存信息之外，还有些关键的程序运行状态也会同时 dump 下来，例如寄存器信息（包括程序指针、栈指针等）、内存管理信息、其他处理器和操作系统状态和信息。core dump 对于编程人员诊断和调试程序是非常有帮助的，因为对于有些程序错误是很难重现的，例如指针异常，而 core dump 文件可以再现程序出错时的情景。

7.5.6.2 开启 coredump

可以使用命令 ulimit 开启，也可以在程序中通过 setrlimit 系统调用开启。

在终端输入命令 `$ ulimit -c` 输出结果为 0，说明默认是关闭 coredump 的，即当程序异常终止时，不会生成 coredump 文件。可以使用 `$ ulimit -c unlimited` 来开启 coredump 功能，并且不限制 coredump 文件大小；如果需要限制文件大小，将 unlimited 改成你想生成 core 文件的大小，注意单位是 KB。

7.5.6.3 调试 core 文件

调试使用 PC 上的 gdb 查看程序挂载位置。但调试开发板程序需要使用交叉编译工具链中的 gdb 命令

glibc:mips-linux-gnu-gdb

uclibc: mips-linux-uclibc-gnu-gdb

使用 `mips-linux-***-gdb [program] [core]` 查看 core 文件，其中 program 是可执行程序文件名，core 是生成的 core 文件名。例如：

```
$ mips-linux-uclibc-gnu-gdb ./build core-build
```

7.5.7 gdbserver 使用方法

gdbserver 是嵌入式开发调试的主要工具，依赖开发板上的 gdbserver 程序以及交叉编译工具链中的 mips-linux-gnu-gdb 命令。gdbserver 需要调试的 PC 和开发板网络相通。

本文仅介绍 gdbserver 的开启及连接方法，相关命令遵循标准 gdbserver 命令，详细内容请在互联网查询。

gdb 以及 gdbserver 命令位置：

A. gdb 在 toolchain 文件夹下，位于 bin/mips-linux-gnu-gdb

B. gdbserver 位于 toolchain 文件夹下：

glibc: mips-linux-gnu/libc/usr/lib/bin/gdbserver

uclibc: mips-linux-gnu/libc/uclibc/usr/lib/bin/gdbserver

1) PC 运行 gdb

```
$ mips-linux-gnu-gdb [PC 端应用程序路径]
```

例如：

```
$ mips-linux-gnu-gdb sample-Encoder-h264
GNU gdb (Ingenic 2015.02) 7.4.50.20120716-cvs
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type “show copying”and
“show warranty”for details.
This GDB was configured as “--host=i686-pc-linux-gnu
--target=mips-linux-gnu”.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from sample-Encoder-h264...done.
(gdb)
```

注：

A. 如果出现<code>(no debugging symbols found)</code>的提示，可能是因为应用程序 strip 过了（PC 端），删除了 debug 需要的段，无法进行 gdb 调试。

B. 开发板上程序是可以 strip 的，Symbol 的读取在 PC 端 gdb 工具完成。这也就是

gdbserver 相对与 local gdb 以及 core dump 的优势——开发板端可以 strip——因为前两者都需要在开发板端 load symbol，因此应用程序会变得非常大，无法放在 flash 中。

2) 设置动态库搜索路径

应用程序往往动态链接 libc、ld 等库，当运行于动态库中 break 时（比如 memcpy）往往无法跟踪到当前 pc 所在位置。这是因为动态库的加载地址是不固定的。因此需要在 PC 端 gdb 上指定动态库加载路径。方法是：

```
set solib-search-path
/opt/mips-gcc472-glibc216-64bit/mips-linux-gnu/libc/lib/:/opt/mips-gcc472-
glibc216-64bit/mips-linux-gnu/lib
```

多个路径之间通过“：”来隔开。注：上述例子中仅指定了 toolchain 中 glibc 的 libc 系列库和 libstdc++等主要外部库。

A. glibc 的动态库在 toolchain 文件夹下的相对目录为：mips-linux-gnu/libc/lib 与 mips-linux-gnu/lib

B. uclibc 的动态库在 toolchain 文件夹下的相对目录为：
mips-linux-gnu/libc/uclibc/lib 与 mips-linux-gnu/lib/uclibc

3) 开发板启动 gdbserver

```
$ gdbserver [PC 机 IP:端口，与步骤 2 中端口需一致] [应用程序路径]
```

例如：

```
$ gdbserver 192.168.1.100:1234 /tmp/sample-Encoder-h264
```

4) 链接开发板

```
$ target remote [开发板 IP:端口]
```

例如：

```
$ target remote 192.168.1.101:1234
```

5) gdbserver 连通，开发板会出现如下信息：

```
(gdb) target remote 192.168.1.101:1234
Remote debugging using 192.168.1.101:1234
Reading symbols from
/opt/mips-gcc472-glibc216-64bit/mips-linux-gnu/libc/lib/ld.so.1...done.
Loaded symbols for
/opt/mips-gcc472-glibc216-64bit/mips-linux-gnu/libc/lib/ld.so.1
0x77fc7070 in __start () from
/opt/mips-gcc472-glibc216-64bit/mips-linux-gnu/libc/lib/ld.so.1
```

至此，即可在 PC 端运行 gdb 命令。

注：与 gdbserver 与 local gdb 不同，启动程序的命令为“c”(continue)。

7.5.8 IMPSDK 调试命令

1) Rmem

rmem 的内存可以使用 SDK 中的 rmem 计算器 excel 表格来计算得到；SDK remem 内存使用信息可以使用下面命令查看：

```
impdbg --sys;logcat
```

2) 查看数据流处理统计

update_cnt 是每个 group 模块线程的处理次数统计 可以通过下面命令来获取 10 秒间隔的统

计来计算处理帧率。

```
[root@IPC:tmp]# ./impdbg --sys;sleep 10;./impdbg --sys
tree item: 3
Framesource-0       update_cnt=81540(qframecnt=81538, dqframecnt=81540,
sem_msg_cnt=16, sem_cnt=0)
OSD-0               update_cnt=81540(sem_msg_cnt=16, sem_cnt=0)
Encoder-0           update_cnt=81540(sem_msg_cnt=16, sem_cnt=0)
---Framesource-0-----OSD-0------------Encoder-0
tree item: 4
Framesource-1       update_cnt=108828(qframecnt=108823,
dqframecnt=108823, sem_msg_cnt=16, sem_cnt=0)
IVS-0               update_cnt=108828(sem_msg_cnt=16, sem_cnt=0)
OSD-1               update_cnt=108828(sem_msg_cnt=16, sem_cnt=0)
Encoder-1           update_cnt=108828(sem_msg_cnt=16, sem_cnt=0)
---Framesource-1-----IVS-0------------OSD-1------------Encoder-1
```

3) 查看编码信息统计

可以通过下面命令获取编码 group，channel 信息，编码类型，码流控制等参数 df 信息，编码通道码流 buf 成功被上次取走的码流统计 df 信息，编码通道码流 buf 慢导致的丢帧数统计。

```
[root@IPC:tmp]# ./impdbg --enc_info;sleep 10;./impdbg --enc_info
GROUP 0
        CHANNEL 0    1920x 1080     START H265 tf:69198      df:6050
encdur:3343131
-------------------------------------------------
ch->index = 0
rcAttr->outFrmRate->frmRateNum = 25(0x19) offset:size = 0:4
rcAttr->outFrmRate->frmRateDen = 1(0x1) offset:size = 4:4
rcAttr->maxGop = 50(0x32) offset:size = 8:4
rcAttr->rcMode->rcMode = 3(0x3) offset:size = 12:4
rcAttr->rcMode->Smart->maxQp = 45(0x2d) offset:size = 16:4
rcAttr->rcMode->Smart->minQp = 15(0xf) offset:size = 20:4
rcAttr->rcMode->Smart->staticTime = 2(0x2) offset:size = 24:4
rcAttr->rcMode->Smart->maxBitRate = 2048(0x800) offset:size = 28:4
rcAttr->rcMode->Smart->iBiasLvl = -3(0xfffffffd) offset:size = 32:4
rcAttr->rcMode->Smart->changePos = 80(0x50) offset:size = 36:4
rcAttr->rcMode->Smart->qualityLvl = 2(0x2) offset:size = 40:4
rcAttr->rcMode->Smart->frmQPStep = 3(0x3) offset:size = 44:4
rcAttr->rcMode->Smart->gopQPStep = 15(0xf) offset:size = 48:4
rcAttr->rcMode->Smart->flucLvl = 2(0x2) offset:size = 52:4
rcAttr->frmUsed->enable = 0(0x0) offset:size = 56:1
rcAttr->frmUsed->frmUsedMode = 0(0x0) offset:size = 60:4
rcAttr->frmUsed->frmUsedTimes = 0(0x0) offset:size = 64:4
```

```
    rcAttr->denoise->enable = 0(0x0) offset:size = 68:1
    rcAttr->denoise->dnType = 0(0x0) offset:size = 72:4
    rcAttr->denoise->dnIQp = 0(0x0) offset:size = 76:4
    rcAttr->denoise->dnPQp = 0(0x0) offset:size = 80:4
    rcAttr->hSkip->hSkipAttr->skipType = 0(0x0) offset:size = 84:4
    rcAttr->hSkip->hSkipAttr->m = 49(0x31) offset:size = 88:4
    rcAttr->hSkip->hSkipAttr->n = 1(0x1) offset:size = 92:4
    rcAttr->hSkip->hSkipAttr->maxSameSceneCnt = 1(0x1) offset:size = 96:4
    rcAttr->hSkip->hSkipAttr->bEnableScenecut = 0(0x0) offset:size = 100:4
    rcAttr->hSkip->hSkipAttr->bBlackEnhance = 0(0x0) offset:size = 104:4
    rcAttr->hSkip->maxHSkipType = 1(0x1) offset:size = 108:4
    GROUP 1
           CHANNEL 1     640x  480     START H265 tf:83485      df:1506
encdur:3343132
    ------------------------------------------------
    ch->index = 1
    rcAttr->outFrmRate->frmRateNum = 25(0x19) offset:size = 0:4
    rcAttr->outFrmRate->frmRateDen = 1(0x1) offset:size = 4:4
    rcAttr->maxGop = 50(0x32) offset:size = 8:4
    rcAttr->rcMode->rcMode = 3(0x3) offset:size = 12:4
    rcAttr->rcMode->Smart->maxQp = 45(0x2d) offset:size = 16:4
    rcAttr->rcMode->Smart->minQp = 15(0xf) offset:size = 20:4
    rcAttr->rcMode->Smart->staticTime = 2(0x2) offset:size = 24:4
    rcAttr->rcMode->Smart->maxBitRate = 546(0x222) offset:size = 28:4
    rcAttr->rcMode->Smart->iBiasLvl = -3(0xfffffffd) offset:size = 32:4
    rcAttr->rcMode->Smart->changePos = 80(0x50) offset:size = 36:4
    rcAttr->rcMode->Smart->qualityLvl = 2(0x2) offset:size = 40:4
    rcAttr->rcMode->Smart->frmQPStep = 3(0x3) offset:size = 44:4
    rcAttr->rcMode->Smart->gopQPStep = 15(0xf) offset:size = 48:4
    rcAttr->rcMode->Smart->flucLvl = 2(0x2) offset:size = 52:4
    rcAttr->frmUsed->enable = 0(0x0) offset:size = 56:1
    rcAttr->frmUsed->frmUsedMode = 0(0x0) offset:size = 60:4
    rcAttr->frmUsed->frmUsedTimes = 0(0x0) offset:size = 64:4
    rcAttr->denoise->enable = 0(0x0) offset:size = 68:1
    rcAttr->denoise->dnType = 0(0x0) offset:size = 72:4
    rcAttr->denoise->dnIQp = 0(0x0) offset:size = 76:4
    rcAttr->denoise->dnPQp = 0(0x0) offset:size = 80:4
    rcAttr->hSkip->hSkipAttr->skipType = 0(0x0) offset:size = 84:4
    rcAttr->hSkip->hSkipAttr->m = 49(0x31) offset:size = 88:4
    rcAttr->hSkip->hSkipAttr->n = 1(0x1) offset:size = 92:4
    rcAttr->hSkip->hSkipAttr->maxSameSceneCnt = 1(0x1) offset:size = 96:4
    rcAttr->hSkip->hSkipAttr->bEnableScenecut = 0(0x0) offset:size = 100:4
```

```
    rcAttr->hSkip->hSkipAttr->bBlackEnhance = 0(0x0) offset:size = 104:4
    rcAttr->hSkip->maxHSkipType = 1(0x1) offset:size = 108:4

    GROUP 0
            CHANNEL 0    1920x 1080     START H265 tf:69319      df:6050
encdur:3353252
    ------------------------------------------------
    ch->index = 0
    rcAttr->outFrmRate->frmRateNum = 25(0x19) offset:size = 0:4
    rcAttr->outFrmRate->frmRateDen = 1(0x1) offset:size = 4:4
    rcAttr->maxGop = 50(0x32) offset:size = 8:4
    rcAttr->rcMode->rcMode = 3(0x3) offset:size = 12:4
    rcAttr->rcMode->Smart->maxQp = 45(0x2d) offset:size = 16:4
    rcAttr->rcMode->Smart->minQp = 15(0xf) offset:size = 20:4
    rcAttr->rcMode->Smart->staticTime = 2(0x2) offset:size = 24:4
    rcAttr->rcMode->Smart->maxBitRate = 2048(0x800) offset:size = 28:4
    rcAttr->rcMode->Smart->iBiasLvl = -3(0xfffffffd) offset:size = 32:4
    rcAttr->rcMode->Smart->changePos = 80(0x50) offset:size = 36:4
    rcAttr->rcMode->Smart->qualityLvl = 2(0x2) offset:size = 40:4
    rcAttr->rcMode->Smart->frmQPStep = 3(0x3) offset:size = 44:4
    rcAttr->rcMode->Smart->gopQPStep = 15(0xf) offset:size = 48:4
    rcAttr->rcMode->Smart->flucLvl = 2(0x2) offset:size = 52:4
    rcAttr->frmUsed->enable = 0(0x0) offset:size = 56:1
    rcAttr->frmUsed->frmUsedMode = 0(0x0) offset:size = 60:4
    rcAttr->frmUsed->frmUsedTimes = 0(0x0) offset:size = 64:4
    rcAttr->denoise->enable = 0(0x0) offset:size = 68:1
    rcAttr->denoise->dnType = 0(0x0) offset:size = 72:4
    rcAttr->denoise->dnIQp = 0(0x0) offset:size = 76:4
    rcAttr->denoise->dnPQp = 0(0x0) offset:size = 80:4
    rcAttr->hSkip->hSkipAttr->skipType = 0(0x0) offset:size = 84:4
    rcAttr->hSkip->hSkipAttr->m = 49(0x31) offset:size = 88:4
    rcAttr->hSkip->hSkipAttr->n = 1(0x1) offset:size = 92:4
    rcAttr->hSkip->hSkipAttr->maxSameSceneCnt = 1(0x1) offset:size = 96:4
    rcAttr->hSkip->hSkipAttr->bEnableScenecut = 0(0x0) offset:size = 100:4
    rcAttr->hSkip->hSkipAttr->bBlackEnhance = 0(0x0) offset:size = 104:4
    rcAttr->hSkip->maxHSkipType = 1(0x1) offset:size = 108:4
    GROUP 1
            CHANNEL 1     640x  480     START H265 tf:83738      df:1506
encdur:3353253
    ------------------------------------------------
    ch->index = 1
    rcAttr->outFrmRate->frmRateNum = 25(0x19) offset:size = 0:4
```

```
rcAttr->outFrmRate->frmRateDen = 1(0x1) offset:size = 4:4
rcAttr->maxGop = 50(0x32) offset:size = 8:4
rcAttr->rcMode->rcMode = 3(0x3) offset:size = 12:4
rcAttr->rcMode->Smart->maxQp = 45(0x2d) offset:size = 16:4
rcAttr->rcMode->Smart->minQp = 15(0xf) offset:size = 20:4
rcAttr->rcMode->Smart->staticTime = 2(0x2) offset:size = 24:4
rcAttr->rcMode->Smart->maxBitRate = 546(0x222) offset:size = 28:4
rcAttr->rcMode->Smart->iBiasLvl = -3(0xfffffffd) offset:size = 32:4
rcAttr->rcMode->Smart->changePos = 80(0x50) offset:size = 36:4
rcAttr->rcMode->Smart->qualityLvl = 2(0x2) offset:size = 40:4
rcAttr->rcMode->Smart->frmQPStep = 3(0x3) offset:size = 44:4
rcAttr->rcMode->Smart->gopQPStep = 15(0xf) offset:size = 48:4
rcAttr->rcMode->Smart->flucLvl = 2(0x2) offset:size = 52:4
rcAttr->frmUsed->enable = 0(0x0) offset:size = 56:1
rcAttr->frmUsed->frmUsedMode = 0(0x0) offset:size = 60:4
rcAttr->frmUsed->frmUsedTimes = 0(0x0) offset:size = 64:4
rcAttr->denoise->enable = 0(0x0) offset:size = 68:1
rcAttr->denoise->dnType = 0(0x0) offset:size = 72:4
rcAttr->denoise->dnIQp = 0(0x0) offset:size = 76:4
rcAttr->denoise->dnPQp = 0(0x0) offset:size = 80:4
rcAttr->hSkip->hSkipAttr->skipType = 0(0x0) offset:size = 84:4
rcAttr->hSkip->hSkipAttr->m = 49(0x31) offset:size = 88:4
rcAttr->hSkip->hSkipAttr->n = 1(0x1) offset:size = 92:4
rcAttr->hSkip->hSkipAttr->maxSameSceneCnt = 1(0x1) offset:size = 96:4
rcAttr->hSkip->hSkipAttr->bEnableScenecut = 0(0x0) offset:size = 100:4
rcAttr->hSkip->hSkipAttr->bBlackEnhance = 0(0x0) offset:size = 104:4
rcAttr->hSkip->maxHSkipType = 1(0x1) offset:size = 108:4
```