

Hichip HCRTOS SDK User Manual

Hichip HCRTOS SDK User Manual

1. 开发环境设置

2. 设置第三方软件的本地下载路径

3. 软件有关设计

3.1 海奇hertos sdk软件框架:

3.2 SDK目录介绍

4. 编译和烧录hertos

4.1 编译SDK

4.2 烧录-with-GDB

5. hertos sdk配置设计

5.1 总配置

5.2 总配置内会指定其它有关不同的配置

5.2.1 DTS的配置

6. 支持的功能

6.1 hertos kernel driver

6.2 hcboot

6.2.1 从bootrom到hcboot

6.2.3 压缩与解压

6.2.4 hertos driver

6.3 本地、网络多媒体播放

6.3.1 ffmpeg接口

6.3.2 audio/video驱动接口

6.3.3 mplayer

6.4 开机show-logo/show-av

7. 有关不同板子的配置

7.1 DDR配置

7.2 调试串口配置

8. 驱动程序头文件

9. GDB调试工具

10. 关于4K解码与输出

11. LCD显示屏配置说明

11.1 屏幕验证配置汇总

11.2 LCD显示屏的快速配置说明

11.3 RGB 与LVDS的配置说明

11.4 MIPI的配置

12. 快速开发指南

12.1 SDK 配置、编译和打包

12.1.1 sdk编译和打包

12.2 无线投屏介绍

12.2.1无线投屏代码介绍

12.2.2 dlna接口函数

12.2.3 miracast接口函数

12.2.4 aircast接口函数

12.2.5 wifi manager接口函数

12.2.6 有线同屏接口函数

12.3 如何开启&关闭&更换boot show logo

12.3.1 如何开启boot show logo

12.3.1.1 如果是未编译过的工程,

12.3.1.2 如果是已经编译过的工程

12.3.2 如何关闭boot show logo

12.3.3 如何更换boot show logo

12.3.4 如何更换.h264的logo

- 12.4 屏幕如何做到旋转。
 - 12.4.1 OSD的旋转
 - 12.4.2 视频的旋转:
 - 12.4.3 一键旋转功能
 - 12.4.4 同屏投屏时如何旋转和设置分辨率
- 12.5 SD卡驱动
- 12.6 LVGL最高支持的帧数配置
- 12.7 如何在demo配置之外, 增加定制化的板级配置
 - 12.7.1 增加一个板级配置
- 12.8 hcRTOS Nor Flash区间的分布
 - 12.8.1 流程介绍
 - 12.8.2 如何增加一个客户可读可写的分区?
 - 12.8.3 如何临时添加一个不支持的flash
- 12.9 uart蓝牙的配置
 - 12.9.1 串口蓝牙
- 12.10 wifi的支持
- 12.11 TP驱动力的集成和测试
- 12.12 hdmi in调试
- 12.13 OSD 图层介绍和分辨率的修改
- 12.14 VIDEO图层的介绍和使用、测试
- 12.15 按键类支持
 - 12.15.1 如何增加ir遥控器
 - 12.15.2 如何添加adc key支持
- 12.16 如何在boot启动阶段拉高或拉低gpio
- 12.17 固件升级
 - 12.17.1 hcfota介绍
 - 12.17.2 hcfota实现原理
 - 12.17.3 hcfota支持的方案
 - 12.17.4 hcfota相关代码介绍
 - 12.17.5 hcfota配置
 - 12.17.6 hcfota调试: app如何调用hcfota接口
- 12.8 如何打开cjc8988?
 - 12.8.1 硬件要求:
 - 12.8.2 软件要求:
 - 12.8.3 测试方法
 - 12.8.4 测试结果
 - 12.8.5 其他类似芯片

13. 常见问题Q&A

- 13.1 checkout到新的branch, 比如从2022.07.y更新到2022.09.y, 编译不过?
- 13.2 B200 不同版本串口RX不一样, 如:
- 13.3 配置好屏幕后, 显示图片异常
- 13.4 提示工具链未安装?
- 13.5 编译出现wget 参数错误?
- 13.6 编译提示hdmirx和usbmirror库找不到?
- 13.7 编译后, 板子没声音出来?
- 13.8 遇到hcrtos的SDK问题, 如何报给原厂?
- 13.9 一代芯片支持nand flash & spi nand吗?

1. 开发环境设置

HCRTOS基于Ubuntu 18.04.5 LTS建立主机端开发环境。桌面版和服务版Ubuntu均可作为开发环境, 由于桌面版自带的工具较为丰富, 建议使用桌面版Ubuntu。

Ubuntu的系统语言需要设置为英文。如果系统语言不是英文, 可以通过下面方式修改

1. 编辑文件 `/etc/default/locale`
2. 修改成英文配置:
`LANG="en_US.UTF-8"`
`LANGUAGE="en_US:en"`
3. 然后重启Ubuntu, 即可恢复为英文的语言环境。

Ubuntu主机安装软件时建议使用华为的源

```
$ sudo cp /etc/apt/sources.list /etc/apt/source.list.bak
```

建立一个新的 `/etc/apt/source.list`文件, 内容如下:

```
# deb cdrom:[Ubuntu 18.04.5 LTS _Bionic Beaver_ - Release amd64 (20200806.1)]/
bionic main restricted

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic main restricted
# deb-src http://cn.archive.ubuntu.com/ubuntu/ bionic main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic-updates main
restricted
# deb-src http://cn.archive.ubuntu.com/ubuntu/ bionic-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic universe
# deb-src http://cn.archive.ubuntu.com/ubuntu/ bionic universe
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic-updates universe
# deb-src http://cn.archive.ubuntu.com/ubuntu/ bionic-updates universe

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic multiverse
# deb-src http://cn.archive.ubuntu.com/ubuntu/ bionic multiverse
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic-updates multiverse
# deb-src http://cn.archive.ubuntu.com/ubuntu/ bionic-updates multiverse

## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic-backports main
restricted universe multiverse
# deb-src http://cn.archive.ubuntu.com/ubuntu/ bionic-backports main restricted
universe multiverse

## Uncomment the following two lines to add software from Canonical's
## 'partner' repository.
```

```
## This software is not part of Ubuntu, but is offered by Canonical and the
## respective vendors as a service to Ubuntu users.
# deb http://archive.canonical.com/ubuntu bionic partner
# deb-src http://archive.canonical.com/ubuntu bionic partner

deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic-security main
restricted
# deb-src http://security.ubuntu.com/ubuntu bionic-security main restricted
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic-security universe
# deb-src http://security.ubuntu.com/ubuntu bionic-security universe
deb http://mirrors.huaweicloud.com/repository/ubuntu/ bionic-security multiverse
# deb-src http://security.ubuntu.com/ubuntu bionic-security multiverse
```

更新源:

```
$ sudo apt update
```

Ubuntu主机环境需要安装如下工具:

```
shell
$ sudo apt install git
$ sudo apt install gcc
$ sudo apt install flex
$ sudo apt install bison
$ sudo apt install gperf
$ sudo apt install make
$ sudo apt install python
$ sudo apt install unzip
$ sudo apt install rar
$ sudo apt install dos2unix
$ sudo apt install swig
$ sudo apt install python-dev
$ sudo apt install python3-dev
$ sudo apt install python3-pip
$ sudo apt install clang-format
$ sudo apt install python3
$ sudo apt install python3-pip
$ sudo apt install python3-setuptools
$ sudo apt install python3-wheel
$ sudo apt install ninja-build
$ sudo apt install rename
$ sudo apt install gdb
$ sudo apt install apache2
$ sudo apt install re2c
$ sudo apt install ctags
$ sudo apt install lzip
$ sudo apt install libncurses-dev
$ sudo apt install tree
$ sudo apt install pkg-config
$ sudo apt install cmake
$ sudo apt install python-pip
$ sudo apt install automake
$ sudo apt install lzop
$ sudo apt install doxygen
$ sudo apt install graphviz
$ sudo apt install libssl-dev
```

```
$ sudo apt install genromfs
$ sudo apt install lzma
$ sudo pip3 install fdt /* 需要安装lzma和fdt工具以完成sdk编译 */
$ sudo apt install texinfo
$ sudo apt install mtools
$ sudo apt-get install mtd-tools
```

海奇的hcartos SDK需要用到一个mips32R1 toolchain，下载后安装到/opt/mips32-mti-elf

下载路径如下：

https://gitlab.hichiptech.com:62443/sw/dl/-/blob/main/Codescape.GNU.Tools.Package.2019.09-03-2.for.MIPS32.MTI.Bare.Metal.Ubuntu-18.04.5.x86_64.tar.gz

注意！！：hichip freeRTOS SDK版本 2022.09.y及之前的版本使用的是：

Codescape.GNU.Tools.Package.**2019.09-03**.for.MIPS32.MTI.Bare.Metal.Ubuntu-18.04.5.x86_64.tar.gz

2022.09.y之后的版本使用的是：

Codescape.GNU.Tools.Package.**2019.09-03-2**.for.MIPS32.MTI.Bare.Metal.Ubuntu-18.04.5.x86_64.tar.gz

```
$ sudo tar -xzf Codescape.GNU.Tools.Package.2019.09-03-2.for.MIPS32.MTI.Bare.Metal.Ubuntu-18.04.5.x86_64.tar.gz -C /opt
$ ls /opt/mips32-mti-elf/2019.09-03-2/bin/ -l
total 126760
-rwxr-xr-x 1 ps ps 5612736 6月 17 09:11 mips-mti-elf-addr2line
-rwxr-xr-x 2 ps ps 5791072 6月 17 09:11 mips-mti-elf-ar
-rwxr-xr-x 2 ps ps 8706848 6月 17 09:11 mips-mti-elf-as
-rwxr-xr-x 2 ps ps 5157136 6月 17 09:38 mips-mti-elf-c++
-rwxr-xr-x 1 ps ps 5562648 6月 17 09:11 mips-mti-elf-c++filt
-rwxr-xr-x 1 ps ps 5150200 6月 17 09:38 mips-mti-elf-cpp
-rwxr-xr-x 1 ps ps 108776 6月 17 09:11 mips-mti-elf-elfedit
-rwxr-xr-x 2 ps ps 5157136 6月 17 09:38 mips-mti-elf-g++
-rwxr-xr-x 2 ps ps 5136928 6月 17 09:38 mips-mti-elf-gcc
-rwxr-xr-x 2 ps ps 5136928 6月 17 09:38 mips-mti-elf-gcc-7.4.0
-rwxr-xr-x 1 ps ps 163320 6月 17 09:38 mips-mti-elf-gcc-ar
-rwxr-xr-x 1 ps ps 163168 6月 17 09:38 mips-mti-elf-gcc-nm
-rwxr-xr-x 1 ps ps 163176 6月 17 09:38 mips-mti-elf-gcc-ranlib
-rwxr-xr-x 1 ps ps 3875864 6月 17 09:38 mips-mti-elf-gcov
-rwxr-xr-x 1 ps ps 3283736 6月 17 09:38 mips-mti-elf-gcov-dump
-rwxr-xr-x 1 ps ps 3535840 6月 17 09:38 mips-mti-elf-gcov-tool
-rwxr-xr-x 1 ps ps 6165440 6月 17 09:11 mips-mti-elf-gprof
-rwxr-xr-x 4 ps ps 7740896 6月 17 09:11 mips-mti-elf-ld
-rwxr-xr-x 4 ps ps 7740896 6月 17 09:11 mips-mti-elf-ld.bfd
-rwxr-xr-x 2 ps ps 5655368 6月 17 09:11 mips-mti-elf-nm
-rwxr-xr-x 2 ps ps 6370296 6月 17 09:11 mips-mti-elf-objcopy
-rwxr-xr-x 2 ps ps 7942520 6月 17 09:11 mips-mti-elf-objdump
-rwxr-xr-x 2 ps ps 5791096 6月 17 09:11 mips-mti-elf-ranlib
-rwxr-xr-x 2 ps ps 2062496 6月 17 09:11 mips-mti-elf-readelf
-rwxr-xr-x 1 ps ps 5600912 6月 17 09:11 mips-mti-elf-size
-rwxr-xr-x 1 ps ps 5597816 6月 17 09:11 mips-mti-elf-strings
-rwxr-xr-x 2 ps ps 6370288 6月 17 09:11 mips-mti-elf-strip
```

如上所示则表示mips32R1工具链安装好。

首次编译hertos SDK，由于SDK会从官网下载一些开源的软件和工具，确保主机开发环境网络联通。

```
$ cd hertos
$ make O=bl hichip_hc15xx_db_b100_v12_hcscreen_bl_defconfig
$ make O=bl all
$ make hichip_hc15xx_db_b100_v12_hcscreen_defconfig
$ make all
```

编译完成后，会在output/images目录下生成可用于GDB download debug或烧录flash的文件。

```
$ tree output/images/
output/images/
├── aux_code_ddr2_64M_1066MHz.abs
├── bootloader.bin
├── dtb.bin
├── flashwr_unify.abs
├── fs-partition1-root
│   └── logo.hc
├── fs-partition2-root
├── fs-partition3-root
├── hc15xx-db-a110.dtb
├── hc15xx_ddr2_64M_1066MHz.abs
├── hcboot.bin
├── hcboot.out
├── hcboot.out.map
├── hcboot.uncompress.bin
├── hcboot.uncompress.map
├── hcboot.uncompress.out
├── hcdemo.bin
├── hcdemo.bin.gz
├── hcdemo.out
├── hcdemo.out.map
├── hcdemo.uImage
├── HCFota_Generator.exe
├── hcprog.ini
├── HCProgram_bridge.exe
├── persistentmem.bin
├── romfs.img
├── sfburn.bin
├── sfburn.bin.d82acd59b3e10ed73c7c79fcc9af185e
├── sfburn.ini
└── updater.bin
```

后续的编译需要按更改的内容进行！

2.设置第三方软件的本地下载路径

hertos SDK在编译过程中会下载一些第三方软件包，一般会从公开网络进行下载。有时候由于网络问题导致下载很慢或者无法下载，则可以用其他方式提前下载好第三方软件包。然后能过hertos的配置选择从已经下载的路径进行三方软件的下载。

比如，将第三方软件包下载到本地路径

```
/media/data/dl
```

hertos sdk默认会尝试从 `http://hichip01/dl` 进行下载，该路径是海奇内部网络建立的http server，外部网络无法连接，从而sdk尝试失败后会从公开网络去下载。当第三方软件包已经下载好之后，可以通过如下方式修改第三方软件的首选下载路径：

```
$ cd herτος
$ make menuconfig
> Mirrors and Download locations
    (file:///media/data/dl) Primary download site
```

通过修改第三方软件的首选下载路径到 `file:///media/data/dl` 以节省三方软件下载时间。

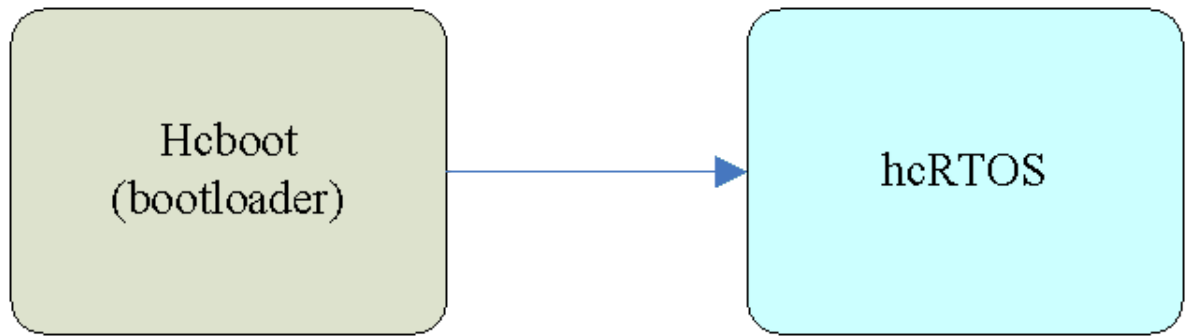
3. 软件有关设计

3.1 海奇hertos sdk软件框架：

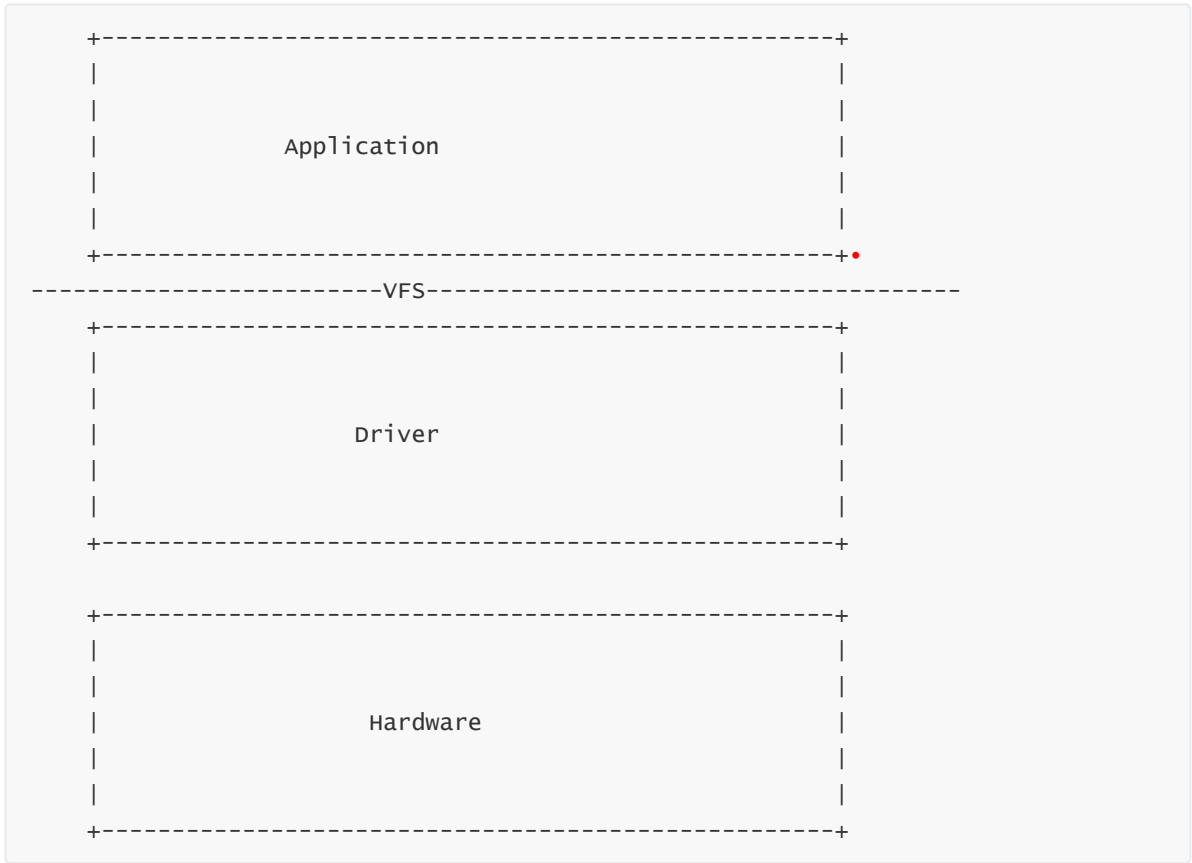
海奇hcRTOS的系统启动采用hcboot。

Hcboot基于hcRTOS开发平台进行开发，经过深度裁减后用于启动hcRTOS系统固件。

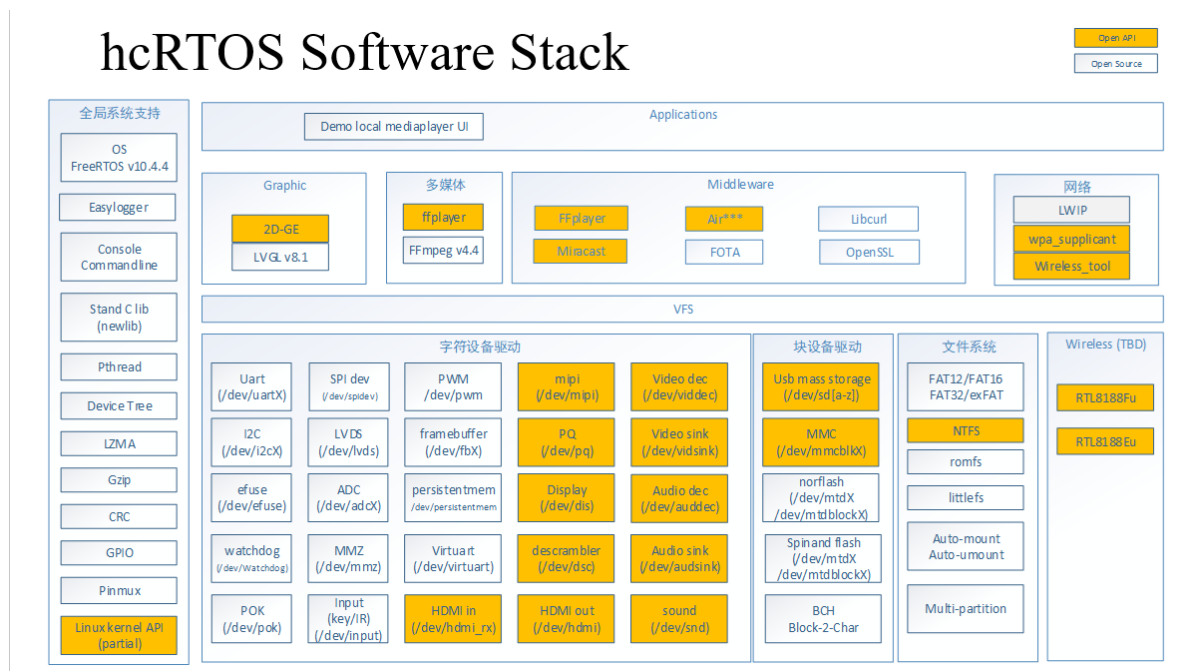
hcRTOS是基于FreeRTOS v10.4.4 内核开发，提供Posix Compatible API。



其简略架构图如下所示：



详细架构图如下所示：



- 驱动程序通过VFS向应用提供posix标准的
`open()` / `close()` / `read()` / `write()` / `poll()` / `ioctl()` 等接口。

3.2 SDK目录介绍

```
finn.wei@hichip01:hcrtos-2022.12.y$ tree -L 2
```

```
.
├── board                                     //板级
├── 配置                                     //海奇
│   ├── hc15xx
│   ├── hc16xx                             //海奇
│   └── hc1xxx
├── build                                     //编译
├── 组织makefile和脚本相关。
│   ├── app-deps.mk
│   ├── download
│   ├── gnuconfig
│   ├── kconfig
│   ├── Kconfig.dummy
│   ├── Makefile.build
│   ├── Makefile.clean
│   ├── Makefile.entry
│   ├── Makefile.in
│   ├── Makefile.link
│   ├── Makefile.rules
│   ├── misc
│   ├── pkg-autotools.mk
│   ├── pkg-cmake.mk
│   ├── pkg-download.mk
│   ├── pkg-generic.mk
│   ├── pkg-release.mk
│   ├── pkg-utils.mk
│   ├── scripts
│   └── tools
```


| | |
|--|----------|
| └─ components | //代码路 |
| 径 | |
| └─ applications | // 上层应 |
| 用。 | |
| └─ bluetooth | //蓝牙驱 |
| 动 | |
| └─ cJSON | |
| └─ cmds | //串口命 |
| 令行测试代码 | |
| └─ dtc | |
| └─ ffmpeg | //媒体播 |
| 放中间件 | |
| └─ hccast | //海奇同 |
| 屏器组件 | |
| └─ hc-examples | //海奇 |
| app单功能测试用例。 | |
| └─ hcfota | // 海奇升 |
| 级组件 | |
| └─ hclib | // 底层库 |
| 文件，未开放代码 | |
| └─ kconfig | |
| └─ kernel | //kernel |
| 驱动组件 | |
| └─ ld | //链接脚 |
| 本 | |
| └─ libcurl | // |
| libcurl相关 | |
| └─ liblvg1 | // lvgl |
| 库代码 | |
| └─ liblzo | // 压缩算 |
| 法代码 | |
| └─ libogg | // ogg库 |
| └─ libopenssl | // |
| openssl相关库代码 | |
| └─ libusb | //usb库 |
| 相关 | |
| └─ libvorbis | // |
| vorbis库 | |
| └─ lzo1x | // lzo压 |
| 缩算法 | |
| └─ newlib | // 标准库 |
| └─ opencore-amr | //amr 库 |
| └─ prebuilts | //海奇预 |
| 编译的库文件 | |
| └─ pthread | |
| └─ quicklz | |
| └─ uboot-tools | |
| └─ zlib | //zlib压 |
| 缩算法 | |
| └─ configs | //芯片配 |
| 置 | |
| └─ hichip_hc15xx_db_a210_hcscreen_b1_defconfig | //a210芯 |
| 片的bootloader配置 | |
| └─ hichip_hc15xx_db_a210_hcscreen_defconfig | //a210的 |
| app配置 | |
| └─ hichip_hc16xx_cb_d3101_v20_projector_c1_cj01_b1_defconfig | //d3101 |
| 的bootloader配置 | |

```

|   └─ hichip_hc16xx_cb_d3101_v20_projector_c1_cj01_defconfig //d3101
的app配置。
└─ d1
|   └─ cJSON
|   └─ dtc
|   └─ libopenssl
|   └─ uboot-tools
|   └─ zlib
└─ document //海奇开
发文档相关
|   └─ 1600 hcRTOS PQ工具使用说明.pdf
|   └─ HCPProgram_i2c_master_userguide.pdf
|   └─ hcrtos-keyadc使用说明.pdf
|   └─ hcrtos-pinmux配置说明.pdf
|   └─ hcrtos_uart_upgrade_user_guide.pdf
|   └─ hcrtos_uart使用说明.pdf
|   └─ hcrtos_user_manual.pdf
|   └─ hcrtos_wifi_user_guide.pdf
|   └─ hichipgdb_userguide.pdf
|   └─ stack_probe_tool_for_debug.pdf
└─ hrepo
└─ kconfig //总配置
文件
└─ Makefile //总
makefile文件
└─ output //最终
app编译中间文件和编译输出文件，默认最终输出文件夹为output
|   └─ build
|   └─ host
|   └─ images
|   └─ include
|   └─ staging
└─ output_b1
//bootloader输出文件。 文件夹名字来自于编译时的O=xxx
|   └─ build
|   └─ host
|   └─ include
|   └─ staging
└─ target
    └─ chip
        └─ kconfig

```

4. 编译和烧录hcrtos

4.1 编译SDK

```

make O=b1 hichip_hc15xx_db_a210_hcscreen_b1_defconfig
make O=b1 all
make hichip_hc15xx_db_a210_hcscreen_defconfig
make all

```

相关常用编译命令：

```
make O=b1 kernel-rebuild all
make O=b1 menuconfig
make kernel-rebuild all
make menuconfig
make apps-hcscreen-rebuild
make hccast-rebuild
...
```

4.2 烧录-with-GDB

编译完成后，会在 `hcartos/output/images` 下生成如下个文件是用于GDB烧录norflash的。

```
$ tree output/images
output/images/
├─ flashwr_unify.abs           // flash初始化文件。
├─ sfburn.bin                 // abs文件，与ini中使用的abs文
                             件一致
├─ sfburn.bin.65501d4f2d86f9dda9abc173271a5773 // ini中使用的abs文件
└─ sfburn.ini                 // gdb烧录时使用的配置文件
```

用GDB工具下载 `hcartos/output/images/sfburn.ini` 到内存，并F5运行。此时，在GDB工具的 **output**窗口的**Mst Output**子选项卡内可以看到烧录norflash的过程，如下所示，直到烧录完成。烧录完成后，重启开发板即可启动系统。

```
Flash size is 1000000
Flash writer: source=0x80060000, target(Offset)=0x00000000, length=0x00280000
Flash size large than 4M, don't do CRC check!
00000000 OK!
00010000 OK!
00020000 OK!
00030000 OK!
00040000 OK!
00050000 OK!
00060000 OK!
00070000 OK!
00080000 OK!
00090000 OK!
000a0000 OK!
000b0000 OK!
000c0000 OK!
000d0000 OK!
000e0000 OK!
000f0000 OK!
00100000 OK!
00110000 OK!
00120000 OK!
00130000 OK!
00140000 OK!
00150000 OK!
00160000 OK!
00170000 OK!
00180000 OK!
00190000 OK!
```

```
001a0000 OK!
001b0000 OK!
001c0000 OK!
001d0000 OK!
001e0000 OK!
001f0000 OK!
00200000 OK!
00210000 OK!
00220000 OK!
00230000 OK!
00240000 OK!
00250000 OK!
00260000 OK!
00270000 OK!
NOR FLASH Burnning Done!
```

更详细的烧录步骤，请参考第9章：GDB调试工具

5. hcrtos sdk配置设计

5.1 总配置

顶层总配置在 `hcrtos/configs` 目录，可以使用 `ls configs` 进行查看。然后根据相应的板子型号选择对应的defconfig

| 配置文件 | 说明 |
|---|-------------|
| ----- ----- | ----- ----- |
| hcrtos/configs/hichip_hc15xx_db_b100_v12_hcdemo_b1_defconfig 的配置文件 | 用于生成hcboot |
| hcrtos/configs/hichip_hc15xx_db_b100_v12_hcdemo_defconfig | 用于生成固件 |

5.2 总配置内会指定其它有关不同的配置

5.2.1 DTS的配置

海奇hcrtos SDK使用device tree进行一些设备驱动的描述。为了便于统一管理，海奇DTS文件统一放在 (`hcrtos/board/hichip/hc1xxx/dts`) 目录

```
CONFIG_CUSTOM_DTS_PATH="$(TOPDIR)/board/hc15xx/common/dts/hc15xx-db-b100-  
hcscreen.dts"
```

hcboot和主固件的DTS通常被指定为同一组配置，分别在 `hichip_hc15xx_db_b100_v12_hcdemo_b1_defconfig` 和 `hichip_hc15xx_db_b100_v12_hcdemo_defconfig` 中指定。

DTS默认被嵌入到固件中。如果修改了DTS，通过如下命令进行重新编译，两个都要编译，因为DTS是boot和固件共用的：

```
$ make O=bl kernel-rebuild all
$ make kernel-rebuild all
```

6. 支持的功能

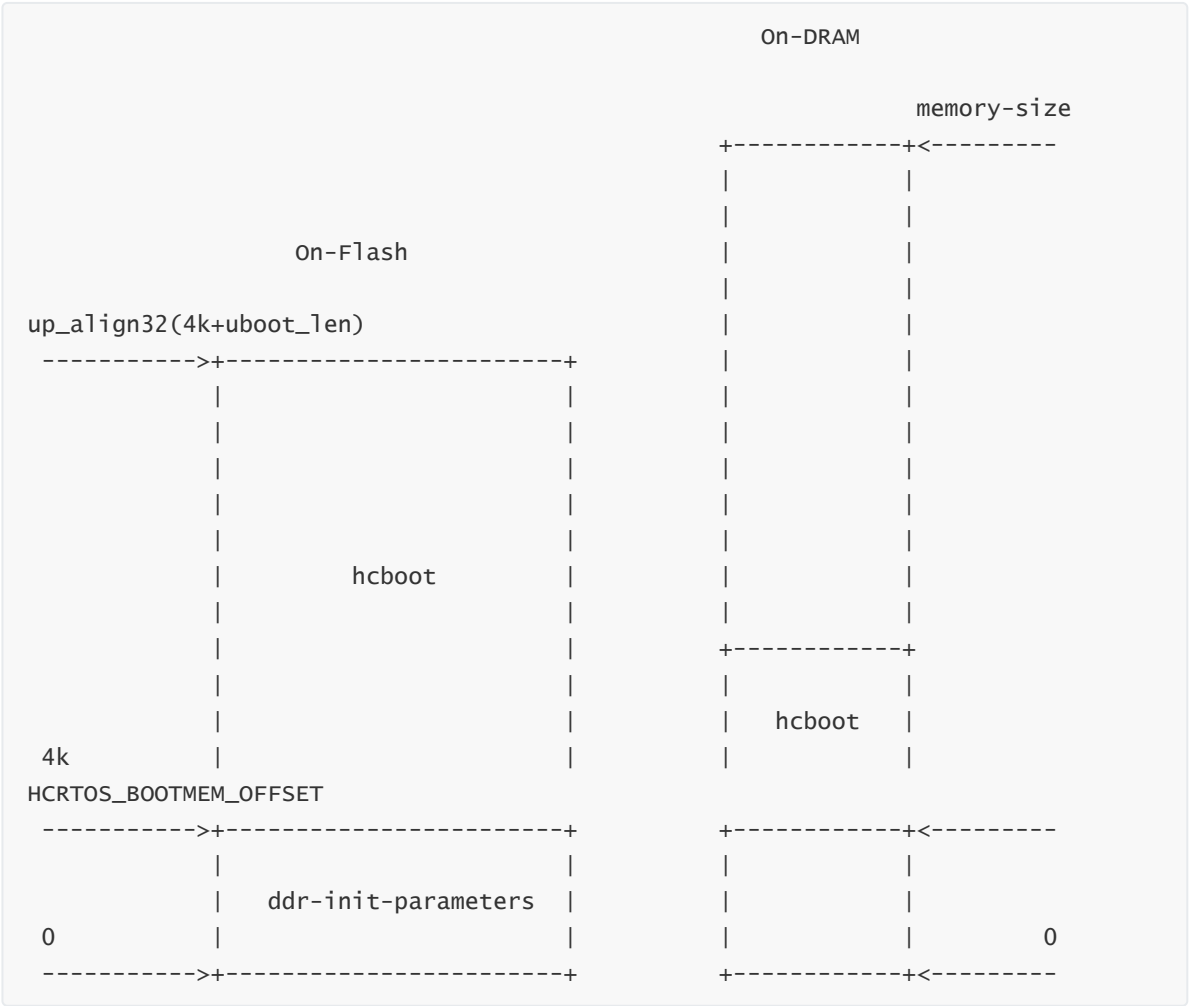
6.1 hcrtos kernel driver

6.2 hcboot

6.2.1 从bootrom到hcboot

hcrtos/output/images/bootloader.bin 文件由hcboot.bin和ddr初始化文件合并而成。

ddr初始化文件位于 board/hichip/hc1xxx/ddrinit 目录，每个文件都是4KB。在norflash上ddr初始化文件与hcboot的mapping方式如下所示。



芯片内部的bootrom程序会从norflash前面4KB获取ddr初始化数据并对内存进行初始化。然后加载hcboot到内存的 HCRTOS_BOOTMEM_OFFSET (该变量在DTS文件中通过宏定义设置) 位置并运行。ddr初始化文件内部除了ddr参数以外，还有其他参数，包括hcboot在norflash上的位置（默认位于4KB位置），hcboot的size，以及hcboot将被加载到内存的哪一个地址运行。由于hcboot的size会经常变化，所以在hcrtos SDK的 post-build.sh 脚本中合并生成 bootloader.bin 时，会动态修改ddr初始化文件并将hcboot的真实size (向上32字节对齐)进行更新。参考脚本：

```
$ cat hcrtos/board/hichip/hc1xxx/post-build.sh
if [ -f ${IMAGES_DIR}/hcboot.out ] && [ -f ${IMAGES_DIR}/hcboot.bin ] && [ -f
${BR2_EXTERNAL_BOARD_DDRINIT_FILE} ]; then

message "Generating bootloader.bin ....."
...
...
message "Generating bootloader.bin done!"
fi
```

6.2.3 压缩与解压

在制作ulmage的时候可以指定不同的压缩方式或者不压缩。参考

```
$ cat hcrtos/board/hc15xx/post-build.sh
if [ -f ${IMAGES_DIR}/${app}.bin ] ; then
    message "Generating ${app}.uImage ....."
    gzip -kf9 ${IMAGES_DIR}/${app}.bin > ${IMAGES_DIR}/${app}.bin.gz
    ${HOST_DIR}/bin/mkimage -A mips -O u-boot -T standalone -C gzip -n ${app}
-e ${app_ep} -a ${app_load} \
    -d ${IMAGES_DIR}/${app}.bin.gz ${IMAGES_DIR}/${app}.uImage
    message "Generating ${app}.uImage done!"
fi
```

从2022/9/* 起，hcrtos支持lzma/lzo/gzip 三种压缩/解压缩方式，可根据需要进行配置。

具体参考：SDK-压缩和自解压设置方法.pdf

6.2.4 hcrtos driver

hcboot和主固件程序均基于hcrtos配置编译而成。其驱动程序是共享的。不同的是底层的audio/video驱动程序以预编译的静态库文件形式提供。由于hcboot要求code size尽可能小，于是底层audio/video预编译库是不一样的。通过指定不同的prebuilt库路径来配置

主程序的预编译库：

```
$ cd hcrtos
$ make menuconfig
> Components
() Prebuilt subdirs /* 指处置空则代表链接hcrtos/components/prebuilts/sysroot里面的库*/
[*] prebuilts --->
    [*] ffplayer
    *- audio driver
    *- video driver
    [*] audio decoder plugins --->
        [*] mp3 dec
        [*] aac dec
        [*] ac3 dec
        [ ] eac3 dec
        [*] adpcm dec
        [*] pcm dec
        [*] flac dec
        [*] vorbis dec
        [*] wma dec
        [*] opus dec
```

```
[*] amr dec
```

hcboot的预编译库，仅支持boot show logo:

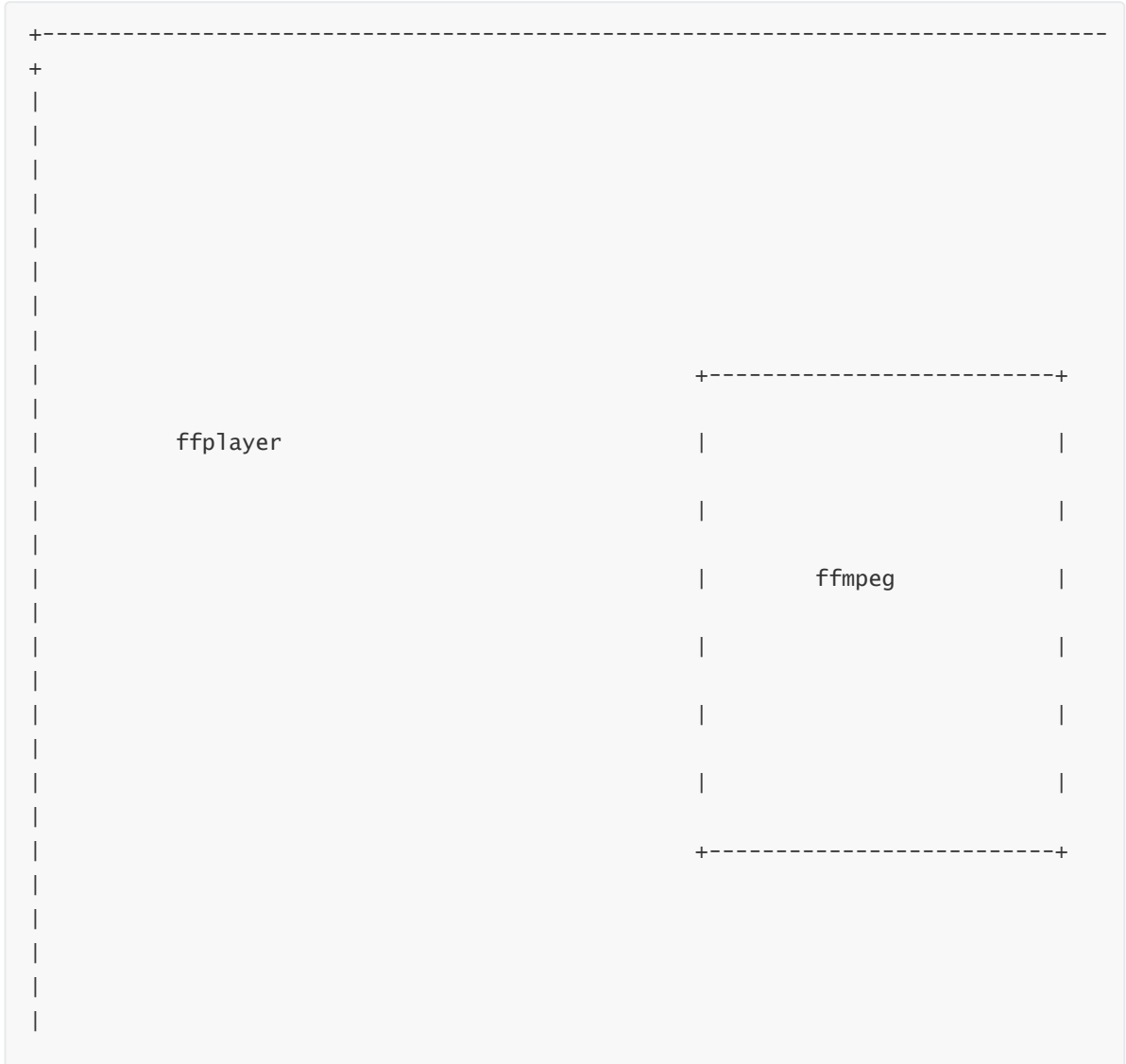
```
$ cd hcrtos
$ make O=output-bl menuconfig
> Components
(boot) Prebuilt subdir /* 代表链接的预编译库路径为
hcrtos/components/prebuilts/boot/sysroot */
[*] prebuilts --->
    [ ]  ffplayer
    [*]  audio driver
    -*  video driver
    [ ]  audio decoder plugins
```

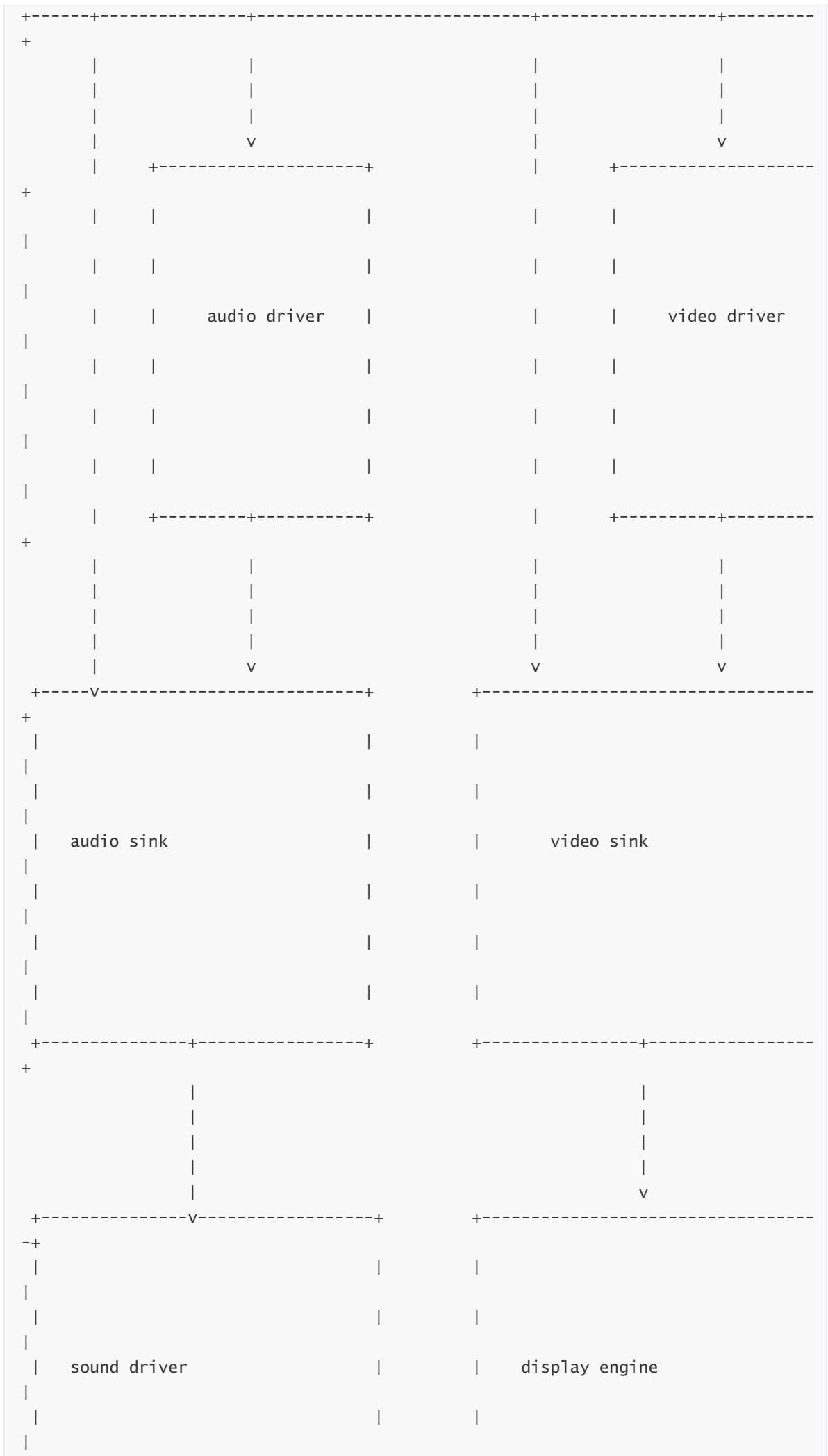
6.3 本地、网络多媒体播放

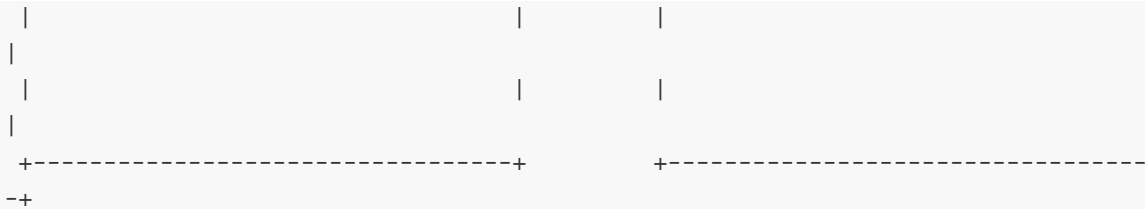
海奇hertos SDK的本地多媒体播放器 (ffplayer) 基于开源软件ffmpeg开发设计。利用ffmpeg中的协议解析, 容器解析以及demux功能获取到audio/video的packat之后, 音视频解码输出与同步则基于AVP提供的audio/video驱动接口实现。

多媒体播放功能的开发即可以基于海奇ffmpeg的接口，也可以基于海奇AVP提供的底层audio/video驱动接口实现。

多媒体音视频驱动框架







- 目前已支持音频多路独立解码输出。最终会利用audio sink模块进行声音混合输出。目前audio sink模块支持resample功能，所以，可支持多路不同samplerate的音频。
 - aac,mp3,vorbis,wma: 不支持多路同类型audio格式同时解码。不同类型audio格式可以同时解码，比如aac 和 mp3同时解码。
 - adpcm,amr,opus,pcm,flac: 以持多路同时解码
- Video不支持多路同时解码
- 多媒体播放的输入接口，比如文件系统等依靠ffmpeg所支持的协议完成。

6.3.1 ffplayer接口

```
$ tree hcrtos/components/prebuilts/
components/prebuilts/
├── boot
│   └── sysroot
│       └── usr
│           ├── include
│           └── lib
│               ├── libaudrv.a
│               └── libvidrv.a
├── kconfig
├── sysroot
│   └── usr
│       ├── include
│       │   ├── ffplayer.h
│       │   ├── glist.h
│       │   ├── opencore-amrnb
│       │   │   ├── interf_dec.h
│       │   │   └── interf_enc.h
│       │   └── opencore-amrwb
│       │       ├── dec_if.h
│       │       └── if_rom.h
│       └── lib
│           ├── aplugin
│           │   ├── libaacdec.a
│           │   ├── libac3dec.a
│           │   ├── libadpcmdec.a
│           │   ├── libamrdec.a
│           │   ├── libeac3dec.a
│           │   ├── libflacdec.a
│           │   ├── libmp3dec.a
│           │   ├── libopencore-amrnb.a
│           │   ├── libopencore-amrwb.a
│           │   ├── libopusdec.a
│           │   ├── libpcmdec.a
│           │   ├── libvorbisdec.a
│           │   └── libwmadec.a
│           ├── libaudrv.a
│           └── libffplayer.a
```

基于ffplayer提供的example位于 `hcrtos/components/hc-examples/source`

6.3.2 audio/video驱动接口

底层audio/video驱动接口的使用方法及调用流程将在未来提供example示范代码。相关的hcuapi接口定义如下：

```
$ ls hcrtos/components/kernel/source/include/uapi/hcuapi/
```

6.3.3 mplayer

hc-examples软件包如果被选上，则会生成mplayer测试命令，它是基于ffplayer所写的一个测试本地多媒体播放的例子程序，供参考。利用mplayer测试命令，可以演示本地多媒体播放功能。

```
hc1600a@dbA3100# mplayer /* 启动mplayer测试程序 */
hc1600a@dbA3100(mp)# play /mnt/usb/test.mkv -s 0 -t 0.1 -d 1 -b 1 /* -s 2表示
不做A/V同步(Free Run); -b 1表示做buffering; -d mode(enum IMG_DIS_MODE)表示图片输出模
式; -t time 从文件的某个时间点开始播放, time < 1时, 为总时间的百分比, 大于1时, 单位为秒*/
hc1600a@dbA3100(mp)# scan /mnt/usb/media /* 表示扫描/mnt/usb/media文件夹, 并
循环播放该文件夹中的媒体文件, 并以audio作为同步参考 */
hc1600a@dbA3100(mp)# scan /mnt/usb/media 0 /* 表示扫描/mnt/usb/media文件夹,
并循环播放该文件夹中的媒体文件, 0表示不做A/V同步 */
hc1600a@dbA3100(mp)# help /* 打印help */
hc1600a@dbA3100(mp)# exit /* 退出mplayer测试命令集 */
```

mplayer内部的其他测试命令可以参考help指令。

6.4 开机show-logo/show-av

默认hcboot所使用的预编译底层a/v驱动只有mpeg2的解码器，没有audio解码器。因此hcboot支持mpeg2格式的视频文件制作而成的logo文件。

类认的logo文件在

```
$ ls hcrtos/board/hc16xx/logo.m2v
board/hc16xx/logo.m2v
$ cat hcrtos/board/hc1xxx/post-build.sh
message "Generating logo.hc ....."
${TOPDIR}/build/scripts/genbootmedia -i ${current_dir}/logo.m2v -o
${IMAGES_DIR}/fs-partition1-root/logo.hc
message "Generating logo.hc done!"
message "Generating romfs.bin ....."
genromfs -f ${IMAGES_DIR}/romfs.img -d ${IMAGES_DIR}/fs-partition1-root/ -v
"romfs"
message "Generating romfs.bin done!"
```

7. 有关不同板子的配置

7.1 DDR配置

ddr配置文件放在 `hcrtos/board/hichip/hc16xx/ddrinit/`, 通过 `menuconfig` 配置选择不同的 ddr 参数文件, 该文件会被 `hcrtos/board/hichip/hc16xx/post-build.sh` 利用并在 `make all` 的时候用于生成最终的 `bootloader.bin`。冷启动的时候 `bootrom` 会利用该配置对 DDR 进行初始化。

```
$ cd hcrtos
$ tree board/hc16xx/common/ddrinit
board/hc16xx/ddrinit/
├─ aux_code_ddr2_128M_1066MHZ.abs
├─ aux_code_ddr2_128M_800MHZ.abs
├─ aux_code_ddr2_64M_800MHZ.abs
├─ aux_code_ddr3_128M_1066MHZ_176Pin.abs
├─ aux_code_ddr3_128M_1066MHZ.abs
├─ aux_code_ddr3_128M_1333MHZ.abs
├─ aux_code_ddr3_128M_1464MHZ.abs
├─ aux_code_ddr3_128M_1560MHZ.abs
├─ aux_code_ddr3_128M_1600MHZ.abs
├─ aux_code_ddr3_128M_800MHZ_176Pin.abs
├─ aux_code_ddr3_128M_800MHZ.abs
├─ aux_code_ddr3_256M_1066MHZ.abs
├─ aux_code_ddr3_256M_1333MHZ.abs
├─ aux_code_ddr3_256M_1600MHZ.abs
├─ aux_code_ddr3_256M_1612MHZ.abs

$ cd hcrtos
$ make menuconfig
> System configuration
($TOPDIR/board/hc16xx/common/ddrinit/aux_code_ddr3_128M_1333MHZ.abs) DDRinit
file
```

用 GDB 进行调试的时候, 需要对 GDB 工具进行配置, 以设置内存初始化参数。GDB 工具所使用的 DDR 参数配置文件在如下路径, 根据不同的板子选择不同的参数文件。

```
$ cd hcrtos
$ tree board/hc16xx/common/ddrinit/gdb/
tree board/hc16xx/common/ddrinit/gdb/
board/hc16xx/common/ddrinit/gdb/
├─ gdb_hc16xx_ddr2_128M_1066MHZ.abs
├─ gdb_hc16xx_ddr2_128M_400MHZ.abs
├─ gdb_hc16xx_ddr2_128M_600MHZ.abs
├─ gdb_hc16xx_ddr2_128M_600MHZ_ODTOff.abs
├─ gdb_hc16xx_ddr2_128M_800MHZ.abs
├─ gdb_hc16xx_ddr2_128M_900MHZ.abs
├─ gdb_hc16xx_ddr3_128M_1066MHZ_176pin.abs
├─ gdb_hc16xx_ddr3_128M_1066MHZ.abs
├─ gdb_hc16xx_ddr3_128M_1200MHZ.abs
├─ gdb_hc16xx_ddr3_128M_1333MHZ.abs
├─ gdb_hc16xx_ddr3_128M_1600MHZ.abs
├─ gdb_hc16xx_ddr3_128M_800MHZ.abs
├─ gdb_hc16xx_ddr3_256M_1066MHZ_176pin.abs
├─ gdb_hc16xx_ddr3_256M_1066MHZ.abs
├─ gdb_hc16xx_ddr3_256M_1200MHZ.abs
├─ gdb_hc16xx_ddr3_256M_1333MHZ.abs
├─ gdb_hc16xx_ddr3_256M_1600MHZ.abs
├─ gdb_hc16xx_ddr3_256M_800MHZ.abs
├─ gdb_hc16xx_sip_ddr3_128M_1333MHZ.abs
├─ gdb_hc16xx_sip_ddr3_128M_1600MHZ.abs
```

```
└─ gdb_hc16xx_sip_ddr3_256M_1333MHz.abs
└─ gdb_hc16xx_sip_ddr3_256M_1600MHz.abs
```

7.2 调试串口配置

以 hc16xx-db-a3100 为例，在 hcrtos/board/hichip/hc16xx/dts/hc16xx-db-a3100.dts 文件中指定了 serial0 设备描述，此处说明使用串口3（四个串口标号为0/1/2/3）

```
$ cat board/hc16xx/dts/hc16xx-db-a3100.dts

uart@3 {
    pinmux = <12 4 13 4>;
    devpath = "/dev/uart3";
    strapin-ctrl-clear = <0x00000000>;
    strapin-ctrl-set = <0x00000000>;
    status = "okay";
};

stdio {
    serial0 = "/hcrtos/uart@3";
};
```

8. 驱动程序头文件

hcrtos SDK的驱动程序都放在 hcrtos/components/kernel/source/include/uapi/hcuapi

驱动程序的头文件可用于应用程序引用开发。

```
.
└─ amprpc.h
└─ auddec.h
└─ audsink.h
└─ avevent.h
└─ avsync.h
└─ codec_id.h
└─ common.h
└─ dis.h
└─ dsc.h
└─ dumpstack.h
└─ dvpdevice.h
└─ efuse.h
└─ fb.h
└─ ge.h
└─ gpio.h
└─ hdmi_rx.h
└─ hdmi_tx.h
└─ i2c-master.h
└─ i2c-slave.h
└─ input-event-codes.h
└─ input.h
└─ iocbase.h
└─ kshm.h
└─ kumsgq.h
└─ lvds.h
└─ mipi.h
```

```
|— mmz.h
|— notifier.h
|— persistentmem.h
|— pinmux
|   |— hc15xx_pinmux.h
|   |— hc16xx_pinmux.h
|— pinmux.h
|— pinpad.h
|— pixfmt.h
|— pq.h
|— pwm.h
|— ramdisk.h
|— rc-proto.h
|— sci.h
|— snd.h
|— spidev.h
|— standby.h
|— sys-blocking-notify.h
|— sysdata.h
|— tvdec.h
|— tvtype.h
|— viddec.h
|— vidmp.h
|— vidsink.h
|— vindvp.h
|— virtuart.h
|— watchdog.h
```

9. GDB调试工具

下载地址: <https://gitlab.hichiptech.com:62443/sw/tools.git>

其中:

\tools\GDB\HiChipGDB-H15\HiChipGDB@H1512 为H15xx系列芯片用的gdb, 其中A210/A110/B200/B100等都为H15xx系列。

\tools\GDB\HiChipGDB-H16\HiChipGDB-0.1.0为H16xx系列芯片用的gdb, 其中A3100/B3100/C3100/D5000等都为H16xx系列。

\tools\GDB\Driver 为jtag调试工具的驱动程序。安装好后, 建议重启电脑使用。

参考文档: hichipgdb_15xx_userguide.pdf & LINUX_HiChipGDB使用手册.pdf (H16系列芯片)

#

10. 关于4K解码与输出

目前, 4K解码需要256MB的内存, 默认 configs/hichip_hc16xx_db_a3100_xxx_4k_defconfig 配置的DDR参数即为256MB的开发板。

目前, 所有配置默认输出都是2k输出。而不论128MB或256MB均可以支持2k输出或4k输出。系统运行起来后, 有如下几种方式切换输出制式

进入命令行后, 直接输入命令

```
4k输出:
hc1600a@dbA3100# dis
hc1600a@dbA3100(dis)# tvsys -c 0 -l 1 -d 1 -t 24 -p 1
```

如果要切换回2k制式输出，则输入命令

```
hc1600a@dbA3100(dis)# tvsys -c 0 -l 1 -d 1 -t 1 -p 1
```

或者通过调用驱动API的方式:

在自己开发的应用程序中调用hcuapi的接口，设置不同的制式输出，hcuapi的接口定义在

```
hcartos/components/kernel/source/include/uapi/hcuapi/dis.h
```

example, 上述方法一的命令就是如下example的实现。

```
static int tvsys_test(int argc, char *argv[])
{
    struct dis_tvsys tvsys = { 0 };
    int fd = -1;
    long cmd = -1;
    int opt;

    opterr = 0;
    optind = 0;

    while ((opt = getopt(argc, argv, "c:l:d:t:p:")) != EOF) {
        switch (opt) {
            case 'c':
                cmd = atoi(optarg);
                break;
            case 'l':
                tvsys.layer = atoi(optarg);
                break;
            case 't':
                tvsys.tvtype = atoi(optarg);
                break;
            case 'd':
                tvsys.distype = atoi(optarg);
                break;
            case 'p':
                tvsys.progressive = atoi(optarg);
                break;
            default:
                break;
        }
    }

    fd = open("/dev/dis", O_WRONLY);
    if (fd < 0) {
        return -1;
    }

    if (tvsys.layer != DIS_LAYER_MAIN && tvsys.layer != DIS_LAYER_AUXP) {
        printf("error layer %d\n", tvsys.layer);
        return -1;
    }

    if (tvsys.distype != DIS_TYPE_SD && tvsys.distype != DIS_TYPE_HD &&
        tvsys.distype != DIS_TYPE_UHD) {
```

```
        printf("error display type %d\n", tvsys.distype);
        return -1;
    }

    switch (cmd) {
    case 0:
        cmd = DIS_SET_TVSYST;
        break;
    case 1:
        cmd = DIS_SET_ZOOM;
        break;
    default:
        break;
    }

    ioctl(fd, cmd, &tvsys);

    close(fd);

    return 0;
}
```

11、LCD显示屏配置说明

11.1 屏幕验证配置汇总

| 配置 | RGB | LVDS | MIPI |
|---|------------------------|---------------------------------|------------------------|
| hichip_hc16xx_db_b300_v10_hcdemo_defconfig | * | 1024×600(VESA)1920×1080(VESA默认) | * |
| hichip_hc16xx_db_b3100_v20_hcdemo_defconfig | 800×480(RGB565 默认) | 1024×600(VESA)1920×1080(VESA) | * |
| hichip_hc16xx_db_b3120_v20_hcdemo_defconfig | 800×480(RGB565 默认) | 1024×600(VESA)1920×1080(VESA) | * |
| hichip_hc16xx_db_c3000_v10_hcdemo_defconfig | * | 1024×600(VESA默认)1920×1080(VESA) | * |
| hichip_hc16xx_db_c300_v10_hcdemo_defconfig | | 1024×600(VESA)1920×1080(VESA) | 1080×1920(默认)1920×1200 |
| hichip_hc16xx_db_c300_v20_hcdemo_defconfig | 800×480(RGB666 默认) | * | 1920×1200 |
| hichip_hc16xx_db_c3100_v20_hcdemo_defconfig | * | * | 1920×1200(默认) |
| hichip_hc16xx_db_c5200_v10_hcdemo_defconfig | * | 1024×600(VESA默认)1920×1080(VESA) | * |
| hichip_hc16xx_db_d3000_v10_hcdemo_defconfig | * | 1024×600(VESA默认)1920×1080(VESA) | * |
| hichip_hc16xx_db_d3100_v10_hcdemo_defconfig | 800×480(RGB565 默认) | 1024×600(VESA)1920×1080(VESA) | * |
| hichip_hc16xx_db_d3100_v20_hcdemo_defconfig | * | 1024×600(VESA 默认) | * |
| hichip_hc16xx_db_d5200_v11_hcdemo_defconfig | 800×480(RGB888 VESA默认) | 1024×6001920×1080(VESA) | 1920×1200 |

11.2 LCD显示屏的快速配置说明

1、前言说明

为了方便快速切换lcd显示屏，目前将所有点亮过的所有lcd显示屏都放在"hcrtos\board\hc16xx\common\dtb\lcd\"的目录下了，需要配置lcd可以在这里进行参考；

2、修改默认配置下的lcd显示屏

例如board\hc16xx\common\dtb\hc16xx-db-d3100-v10.dts默认配置的是800×480的RGB显示屏，如果需要改成1024×600的LVDS显示屏，可以做以下修改：

```
// #include "lcd_rgb_800_480_rgb565.dtsi" /*注释掉默认的配置*/  
#include "lcd_lvds_1024_600_vesa.dtsi" /*添加新的lcd显示屏配置*/
```

3、编译

```
cd hcartos  
make O=b1 kernel-rebuild all  
make kernel-rebuild all
```

注意：2022年11月份后的版本才支持

11.3 RGB 与LVDS的配置说明

- 步骤一：配置demo板

配置rgb编译可以参考配置

```
cd hcartos  
rm -rf output*  
rm -rf b1  
make O=b1 hichip_hc16xx-db-d5200_v11_hcdemo_b1_defconfig  
make O=b1 all  
make hichip_hc16xx-db-d5200_v11_hcdemo_defconfig  
make all
```

配置lvds可以参考配置

```
cd hcartos  
rm -rf output*  
rm -rf b1  
make O=b1 hichip_hc16xx-db-d3100_v20_hcdemo_b1_defconfig  
make O=b1 all  
make hichip_hc16xx-db-d3100_v20_hcdemo_defconfig  
make all
```

- 步骤二：查看menuconfig是否打开了lvds驱动(默认打开)

lvds和rgb都是打开CONFIG_DRV_LVDS的配置

1、查看bootloader是否配置了lvds

```
make O=b1 menuconfig  
cd hcartos  
x There is no help available for this option.  
x Symbol: CONFIG_DRV_LVDS [=y]  
x Type : bool  
x Prompt: lvds  
x Location:  
x -> Components  
x -> kernel (BR2_PACKAGE_KERNEL [=y])  
x -> Drivers  
x Defined at drivers:165
```



```
x Depends on: BR2_PACKAGE_KERNEL [=y]
```

2、查看kernel是否配置了lvds

```
make menuconfig
```

```
x There is no help available for this option.
```

```
x Symbol: CONFIG_DRV_LVDS [=y]
```

```
x Type : bool
```

```
x Prompt: lvds
```

```
x Location:
```

```
x -> Components
```

```
x -> kernel (BR2_PACKAGE_KERNEL [=y])
```

```
x -> Drivers
```

```
x Defined at drivers:165
```

```
x Depends on: BR2_PACKAGE_KERNEL [=y]
```

• 步骤三：配置DE

例如 board\hc16xx\common\dts\lcd\lcd_rgb_800_480_rgb888.dtsi

```
&DE {
```

```
    tvtype = <15>; //1. 一般配置为15
```

```
    VPInitInfo {
```

```
        rgb-cfg{
```

```
            b-rgb-enable = <1>; //2、使能RGB设置为 1
```

```
            /*
```

```
            * 0: MODE_PRGB
```

```
            * 1: MODE_SRGB
```

```
            */
```

```
            rgb-mode = <0>; //3、设置为并行模式
```

```
            /*
```

```
            * 0: MODE_PRGB_888_10bit
```

```
            * 1: MODE_PRGB_666
```

```
            */
```

```
            prgb-mode = <0>; //4、设置为RGB888的模式
```

```
            lcd-width = <800>; //5、lcd-width与h-active-len需要一致
```

```
            b-disable-ejtag = <1>;
```

```
            b-dlpc-enale = <0>;
```

```
            timing-para {
```

```
                /* bool type, 0 or 1*/
```

```
                b-enable = <1>; //6、使能为，使能之后才能配置以下的参数
```

```
                /*
```

```
                * VPO_RGB_CLOCK_27M = 1,
```

```
                * VPO_RGB_CLOCK_33M = 2,
```

```
                * VPO_RGB_CLOCK_49M = 3,
```

```
                * VPO_RGB_CLOCK_66M = 4,
```

```
                * VPO_RGB_CLOCK_74M = 5,
```

```
                * VPO_RGB_CLOCK_85M = 6,
```

```
                * VPO_RGB_CLOCK_108M = 7,
```

```
                * VPO_RGB_CLOCK_6_6M = 8,
```

```
                * VPO_RGB_CLOCK_9M = 9,
```

```
                * VPO_RGB_CLOCK_39_6M = 10,
```

```
                * VPO_RGB_CLOCK_74_25M = 11, //H1600
```

```
                * VPO_RGB_CLOCK_148_5M = 12, //H1600
```

```
                */
```

```
            output-clock = <1>; //7、output-clock = h-total-len * v-total-len
```

* 频率（50~60H）

8、按照屏厂提供参数填写相关屏参

```

        h-total-len = <1026>;//h-total-len = h-active-len + h-front-len
+   h-sync-len+h-back-len
            v-total-len = <525>;//v-total-len = v-active-len + v-front-len +
v-sync-len+v-back-len


        h-active-len = <800>;
        v-active-len = <480>;


        h-front-len = <210>;
        h-sync-len = <6>;
        h-back-len = <10>;


        v-front-len = <22>;
        v-sync-len = <3>;
        v-back-len = <20>;
        /* bool type, 0 or 1*/
        h-sync-level = <0>;
        /* bool type, 0 or 1*/
        v-sync-level = <0>;
        frame-rate = <500000>;//9、设置帧率 50HZ
    };
};
};
};
```

- 步骤四：模式设置为RGB&LVDS

- 4.1 配置RGB

1、lvds使能

board\hc16xx\common\dtb\hc16xx-db-d5200-v11.dts

```
lvds: lvds@0xb8860000 {
```

```
lcd-backlight-gpios-rtos = <PINPAD_L15 GPIO_ACTIVE_HIGH>;//设置背光, show logo后打  
开, 高电平有效
```

```
status = "okay";
```

 $\}.$

2、配置lvds节点内容

例如board\hc16xx\common\dts\lcd\lcd_rgb_800_480_rgb888.dtsi //配置rgb565 linux的节点

```
&lvs {
```

```
reg = <0xB8860000 0x200>, <0xB8800000 0x500>;
```

/*

* "lvds" : LVDS screen

- * "rgb888" : RGB screen

- * "rgb666" : RGB screen

- * "rgb565" : RGB screen

```
* "i2so" : I2S OUT
```

- * "gpio" : only GPIO out

/

```
lvds_ch0-type = "rgb888";
```

```
lvds_ch1-type = "rgb888";
```

/

* rgb ggb bgb

* rrb rgb rbb

* rgr rgg rgb

/

```

    rgb_src_sel = "rgb";
    lcd-backlight-gpios-rtos = <PINPAD_L15 GPIO_ACTIVE_HIGH>;//背光
    /
    * 0: E_SRC_SEL_FXDE = 0
    * 1: E_SRC_SEL_4KDE
    * 2: E_SRC_SEL_HDMI_RX
    /
    src_sel = <CONFIG_DE4K_OUTPUT_SUPPORT>;
    pinmux-rgb888 = <PINPAD_B00 1 PINPAD_B01 1 PINPAD_B02 1 PINPAD_B03 1 PINPAD_B04 1
PINPAD_B05 1 PINPAD_B06 1 PINPAD_B07 1>;/RGB888 other pin set/
    pinmux-rgb666 = <PINPAD_B04 4 PINPAD_B07 4>; /RGB666 other pin set*/
    status = "okay";
};

```

• 4.2 配置lvds

1、使能lvds

board\hc16xx\common\dts\hc16xx-db-d3100-v20.dts

```

lvds: lvds@0xb8860000 {
    status = "okay";
};

```

2、配置lvds节点的内容

board\hc16xx\common\dts\lcd\lcd_lvds_1024_600_vesa.dtsi //配置lvds vesa的节点

```

&lvds {
    /*
    * "lvds"    : LVDS screen
    * "rgb888"   : RGB screen
    * "rgb666"   : RGB screen
    * "rgb565"   : RGB screen
    * "i2so"    : I2S OUT
    * "gpio"    : only GPIO out
    */
    lvds_ch0-type = "lvds";//1、设置输出的通道 LVDS D0~D3
    lvds_ch1-type = "lvds";//    LVDS D4~D7

```

```

    /*
    * 0: E_CHANNEL_MODE_SINGLE_IN_SINGLE_OUT ,
    * 1: E_CHANNEL_MODE_SINGLE_IN_DUAL_OUT
    */
    channel-mode = <0>;//2、输出
    /*
    * 0: E_MAP_MODE_VESA_24BIT ,
    * 1: E_MAP_MODE_VESA_18BIT_OR_JEDIA
    */
    map-mode = <0>;//3、设置屏的信号格式 0:VESA 1:JEDIA
    ch0-src_sel = <0>;
    ch1-src_sel = <0>;
    ch0-invert_clk_sel = <0>;
    ch1-invert_clk_sel = <0>;
    ch0-clk_gate = <0>;
    ch1-clk_gate = <0>;
    hsync-polarity = <1>;//与 de-engine 的h-sync-level 保持一致
    vsync-polarity = <1>;//与 de-engine 的v-sync-level 保持一致
    /*

```

```

* 0: E_ADJUST_MODE_FRAME_START
* 1: E_ADJUST_MODE_HSYNC_POS
* 2: E_ADJUST_MODE_VSYNC_POS
*/
even-odd-adjust-mode = <0>;
even-odd-init-value = <0>;

/*
* 0: E_SRC_SEL_FXDE = 0
* 1: E_SRC_SEL_4KDE
* 2: E_SRC_SEL_HDMI_RX
*/
src_sel = <CONFIG_DE4K_OUTPUT_SUPPORT>;
status = "okay";

```

```
};
```

- 步骤五：编译
- 5.1 重新编译

```

cd hcartos
hcartos$ make O=b1 kernel-rebuild all; make kernel-rebuild all

```

- 步骤六：注意事项：
- 6.1 dts注意

假设 board\hc16xx\common\dts\hc16xx-db-d5200-v11.dts 配置了mipi: dsi0, 请将status = "disable", 并重新编译:

```

cd hcartos
hcartos$ make O=b1 kernel-rebuild all; make kernel-rebuild all

```

- 6.2 lvds通道的说明

正确的配置，通道D0~D3可以配置成 "lvds","i2so","gpio",D4~D7可以配置成 "lvds","i2so","gpio"
例如正确的配置：

```

lvds_ch0-type = "lvds"
lvds_ch1-type = "gpio"

lvds_ch0-type = "i2so"
lvds_ch1-type = "lvds"

lvds_ch0-type = "i2so"
lvds_ch1-type = "i2so"

```

错误的配置：

不能够出现 rgb565 rgb666 rgb888 gpio i2so 同时组合的配置，例如：

```
lvds_ch0-type = "rgb565";//rgb666 rgb888
lvds_ch1-type = "gpio";

lvds_ch0-type = "rgb565";//rgb666 rgb888
lvds_ch1-type = "i2so";

lvds_ch0-type = "gpio";
lvds_ch1-type = "i2so";
```

11.4 MIPI的配置

- 步骤一：板级参考配置

MIPI的相关配置只能在hcartos里面进行配置

```
cd hcartos
rm -rf output*
make O=b1 hichip_hc16xx_db_c300_v10_hcdemo_b1_defconfig
make all
cd hcartos
make hichip_hc16xx_db_c300_v10_hcdemo_defconfig
make all
```

- 步骤二：查看menuconfig是否配置了mipi

```
cd hcartos
1、查看bootloader是否配置了mipi
make O=b1 menuconfig
x There is no help available for this option.
  x Symbol: CONFIG_DRV_MIPI [=y]
  x Type   : bool
  x Prompt: mipi
  x Location:
  x   -> Components
  x     -> kernel (BR2_PACKAGE_KERNEL [=y])
  x       -> Drivers
  x Defined at drivers:177
  x Depends on: BR2_PACKAGE_KERNEL [=y]

2、查看kernel是否配置了mipi
make menuconfig
x There is no help available for this option.
  x Symbol: CONFIG_DRV_MIPI [=y]
  x Type   : bool
  x Prompt: mipi
  x Location:
  x   -> Components
  x     -> kernel (BR2_PACKAGE_KERNEL [=y])
  x       -> Drivers
  x Defined at drivers:177
  x Depends on: BR2_PACKAGE_KERNEL [=y]
```

- 步骤三：配置DE

```
board\hc16xx\common\dts\lcd\lcd_mipi_ILI7807D_1080_1920_rgb888.dtsi
```

```

&DE {
    tvtype = <15>;//1.一般为15
    VPInitInfo {
        rgb-cfg{
            b-rgb-enable = <1>;//2、使能RGB设置为 1
            /*
                * 0: MODE_PRGB
                * 1: MODE_SRGB
            */
            rgb-mode = <0>;//3、设置为并行模式
            /*
                * 0: MODE_PRGB_888_10bit
                * 1: MODE_PRGB_666
            */
            prgb-mode = <0>;//4、设置为RGB888的模式
            lcd-width = <1080>;//5、lcd-width与h-active-len需要一致
            screen_timing: timing-para {
                /* bool type, 0 or 1*/
                b-enable = <1>;//6、使能为，使能之后才能配置以下的参数
                /*
                    * VPO_RGB_CLOCK_27M = 1,
                    * VPO_RGB_CLOCK_33M = 2,
                    * VPO_RGB_CLOCK_49M = 3,
                    * VPO_RGB_CLOCK_66M = 4,
                    * VPO_RGB_CLOCK_74M = 5,
                    * VPO_RGB_CLOCK_85M = 6,
                    * VPO_RGB_CLOCK_108M = 7,
                    * VPO_RGB_CLOCK_6_6M = 8,
                    * VPO_RGB_CLOCK_9M = 9,
                    * VPO_RGB_CLOCK_39_6M = 10,
                    * VPO_RGB_CLOCK_74_25M = 11,//H1600
                    * VPO_RGB_CLOCK_148_5M = 12,//H1600
                */
                output-clock = <12>;//7、output-clock = h-total-len * v-total-len
            }
        }
    }
}

* 频率 (50~60H)

8、按照屏厂提供参数填写相关屏参

h-total-len = <1315>;
v-total-len = <1980>;

h-active-len = <1080>;
v-active-len = <1920>;

h-front-len = <200>;
h-sync-len = <5>;
h-back-len = <30>;

v-front-len = <30>;
v-sync-len = <10>;
v-back-len = <20>;
/* bool type, 0 or 1*/
h-sync-level = <1>;
/* bool type, 0 or 1*/
v-sync-level = <1>;
frame-rate = <600000>;//9、设置帧率 60HZ
};
};
};
};

```

- 步骤四：配置mipi

1、使能mipi

```
board\hc16xx\common\dts\hc16xx-db-c300-v10
```

```
mipi: dsi0 {
```

```
    //1、配置gpio
```

```
    enable-gpios=<PINPAD_L11 0 1>;//设置 GPIO L11引脚，设置为GPIO输出模式，高电平起作用
```

```
    reset-gpios=<PINPAD_L00 0 0>;//设置 GPIO L0引脚，设置为GPIO输出模式，低电平起作用
```

```
    //2、设置延时
```

```
    init-delay-ms = <20>;//初始化enable-gpios reset-gpios引脚之后，输出均为低，然后延时20毫秒，之后
```

```
    enable-delay-ms = <120>;//将enable-gpios 引脚使能有效值（本设置拉高），然后延时120毫秒，之后
```

```
    reset-delay-ms = <120>;//将reset-gpios 引脚使能有效值（本设置拉低），然后延时120毫秒，然后将复位引脚释放（本设置拉高），之后
```

```
    prepare-delay-ms = <120>;//延时120毫秒，开始发送命令
```

```
    status = "okay";
```

```
};
```

2、配置mipi节点的内容

```
board\hc16xx\common\dts\lcd\lcd_mipi_ILI7807D_1080_1920_rgb888.dtsi
```

```
&mipi {
```

```
    reg = <0xb884A000 0x300>, <0xb8800444 0x8>, <0xb8800080 0x8>;
```

```
    /*
```

```
    * MIPI_COLOR_RGB565_0 = 0
```

```
    * MIPI_COLOR_RGB565_1 = 1
```

```
    * MIPI_COLOR_RGB565_2 = 2
```

```
    * MIPI_COLOR_RGB666_0 = 3
```

```
    * MIPI_COLOR_RGB666_1 = 4
```

```
    * MIPI_COLOR_RGB888 = 5
```

```
    */
```

```
    dsi,format = <5>;//3、设置输出的格式
```

```
    dsi,lanes = <4>;// 1= <dsi,lans <=4 //4、设置lane数
```

```
    /*
```

```
    * bit 6:5 MIPI_Virtual_Channel_Generic_ID
```

```
    * bit 4 :MIPI_CRC
```

```
    * bit 3 :MIPI_ECC
```

```
    * bit 2 :MIPI_Bus_Turn_request.
```

```
    * bit 1 :MIPI_EOTp
```

```
    * bit 0 :MIPI_EOTp
```

```
    */
```

```
    dsi,cfg = <0x1c>;
```

```
    /*
```

```
    * 0: E_SRC_SEL_FXDE = 0
```

```
    * 1: E_SRC_SEL_4KDE
```

```
    * 2: E_SRC_SEL_HDMI_RX
```

```
    */
```

```
    src_sel = <CONFIG_DE4K_OUTPUT_SUPPORT>;
```

```
    clock-frequency = <0>; //5、输出时钟，大于0有效；等于0，默认会自动计算
```

```
    //一般 clock-frequency = h-total-len x v-total-
```

```
len x 3 x 8 x (frame-rate\1000)HZ / dsi,lanes
```

```
    status = "okay";
```

```
};
```

- 步骤五: mipi初始化配置

```
board\hc16xx\common\dts\lcd\lcd_mipi_ILI7807D_1080_1920_rgb888.dtsi
```

```
&mipi{
panel-init-sequence = [
    39 00 04 FF 78 07 01
    15 00 02 42 11
    .....
    05 78 01 11
    05 78 01 29
];
};
```

5.1 初始化配置说明:

```
15 00 02 80 77
```

```
| | | | |
| | | | 数据
| | | 数据
| | 数据长度
| 延时
```

命令类型 (0x05: 单字节数据 0x15: 双字节数据 0x39: 多字节数据)

5.2 单字节数据举例:

```
05 78 01 29
```

5.3 双字节数据举例:

```
15 00 02 42 11
```

5.4 多字节数据举例:

```
39 00 04 FF 78 07 01
```

- 步骤六: 重新编译

```
cd hcrtos
```

```
hcrtos$ make O=bl kernel-rebuild all; make kernel-rebuild all
```

- 步骤七: 注意事项:

7.1 假如board\hc16xx\common\dts\hc16xx-db-c300-v10配置了lvds, 可将status = "disable"(这只是关闭lvds的功能, C300芯片可以实现LVDS与MIPI双屏显示, 但是需要显示屏的相关屏参是一致才能让显示效果一致), 重新编译

```
cd hcrtos
```

```
hcrtos$ make O=bl kernel-rebuild all; make kernel-rebuild all
```

12. 快速开发指南

12.1 SDK 配置、编译和打包

12.1.1 sdk编译和打包

```
cd hcrtos
make O=b1 hichip_hc15xx_db_b200_v10_b1_defconfig
make O=b1 all
make hichip_hc15xx_db_b200_v10_defconfig //选择方案配置
make all
```

如果一个sdk要编译多个方案板，可以在编译时增加制定输出目录：

```
make O=b1/b200 hichip_hc15xx_db_b200_v10_b1_defconfig
make O=b1/b200 all
make O=output/b200 hichip_hc15xx_db_b200_v10_defconfig
make O=output/b200 all
```

编译后，可以进入到制定的方案目录，进行单个命令的编译

```
cd output/b200
make liblvg1-rebuild all
```

也可以在hcrtos目录直接编译，但就要加上O=xxx：

```
make O=output/b200 liblvg1-rebuild all
```

如果想清除之前的编译，重新编译sdk，请执行如下命令：

```
cd hcrtos
rm -rf output/d3100
或者
make distclean //慎用，清除所有编译产生的过程文件和目标文件，会造成下次编译时重新下载第
三方软件包，
//从而造成编译非常慢
```

如果想编译某一个软件包后，可以使用如下命令编译整个sdk：

```
以lvg1为例：
make liblvg1-rebuild //重新编译lvg1，
make all //sdk编译和打包

或者
make liblvg1-rebuild all //重新编译lvg1，并且编译整个sdk，并且打包

注意：只是执行make all，是不会重新编译lvg1的。
```

12.2 无线投屏介绍

12.2.1无线投屏代码介绍

```
components/applications/apps-hcscreen/ //无线投屏app
components/prebuilts/sysroot/usr/lib/ //对应无线有线的库文件
components/hccast/
└─ hccast.mk
```

```

├─ kconfig
├─ source
│   ├── aircast                //aircast中间件
│   ├── app                    //hccast测试app,hccast_simple_wireless
│   ├── CMakeLists.txt
│   ├── common                //通用封装层和api
│   ├── dlna                  //dlna中间件
│   ├── httpd                 //网页配网代码
│   ├── inc                   //hccast头文件
│   ├── miracast              //miracast中间件
│   ├── udhcp                 //分配ip地址, 开关udhcpd
│   ├── um                    // 有线同屏代码
│   └─ wifi_mgr               //wifi manager, wifi配网、状态等

```

12.2.2 dlina接口函数

位于: components\hccast\source\inc\hccast_dlna.h

```

hccast_dlna_event_callback    //回调函数
int hccast_dlna_service_init(hccast_dlna_event_callback func); //初始化
int hccast_dlna_service_uninit(); //注销
int hccast_dlna_service_start(); //开启dlina服务
int hccast_dlna_service_stop(); //关闭dlina服务

```

12.2.3 miracast接口函数

位于: components\hccast\source\inc\hccast_mira.h

```

int hccast_mira_service_start(); //开启miracast服务
int hccast_mira_service_stop(); //关闭miracast服务
int hccast_mira_player_init(); //初始化播放器
int hccast_mira_get_stat(void); //获得状态
int hccast_mira_service_init(hccast_mira_event_callback func); //初始化
int hccast_mira_service_uninit(); //注销

```

12.2.4 aircast接口函数

位于: components\hccast\source\inc\hccast_air.h

```

int hccast_air_ap_mirror_stat(void);
int hccast_air_ap_audio_stat(void);
int hccast_air_service_init(hccast_air_event_callback aircast_cb);
int hccast_air_service_start();
int hccast_air_service_stop();
void hccast_air_mdnsd_start();
void hccast_air_mdnsd_stop();
void hccast_air_mediaplayer_2_aircast_event(int type, void *param);
int hccast_air_ap_get_mirror_frame_num(void);
int hccast_air_service_is_start();

```

12.2.5 wifi manager接口函数

位于: components\hccast\source\inc\hccast_wifi_mgr.h

12.2.6 有线同屏接口函数

位于: components\hccast\source\inc\hccast_um.h

```
int hccast_um_init();
int hccast_um_deinit();
int hccast_um_param_set(hccast_um_param_t *param);
int hccast_iu_init(hccast_um_cb event_cb);
int hccast_iu_start(char *uuid, hccast_um_cb event_cb);
int hccast_iu_stop();
int hccast_iu_stop_mirroring();
int hccast_aum_start(hccast_aum_param_t *param, hccast_um_cb event_cb);
int hccast_aum_stop();
int hccast_aum_stop_mirroring();
```

12.3 如何开启&关闭&更换boot show logo

hcartos的logo文件，默认放置于路径 \board\hc15xx\common\，命名为 logo.m2v。

在编译app时，会通过 post-build.sh 文件将 logo.m2v 转化成 /fs-partition1-root/logo.hc。

最终以文件的形式存储在 romfs.bin 中，启机时，mount 路径为 /etc/logo.hc。

如果开启了boot show logo，那么在bootloader中，会通过以下代码 @main.c 显示logo：

```
#if defined(CONFIG_BOOT_SHOWLOGO) && !defined(CONFIG_DISABLE_MOUNTPOINT)
    ret = get_mtdblock_devpath(devpath, sizeof(devpath), "eromfs");
    if (ret >= 0)
        ret = mount(devpath, "/etc", "romfs", MS_RDONLY, NULL);

    if (ret >= 0) {
        showlogo(2, ((char *[]){ "showlogo", "/etc/logo.hc" }));
        wait_show_logo_finish_feed();
    }
#endif
```

12.3.1 如何开启boot show logo

以 HC1512@A210 为例：

12.3.1.1 如果是未编译过的工程，

1. 需要在 configs/hichip_hc15xx_db_a210_v12_hcdemo_defconfig 中配置如下：

```
BR2_EXTERNAL_BOOTMEDIA_FILE="${TOPDIR}/board/hc15xx/common/logo.m2v"
```

configs/hichip_hc15xx_db_a210_v12_hcdemo_b1_defconfig 中不需要配置

```
BR2_EXTERNAL_BOOTMEDIA_FILE。
```

2. 修改文件: configs/hichip_hc15xx_db_b200_b1_defconfig，打开以下句子：

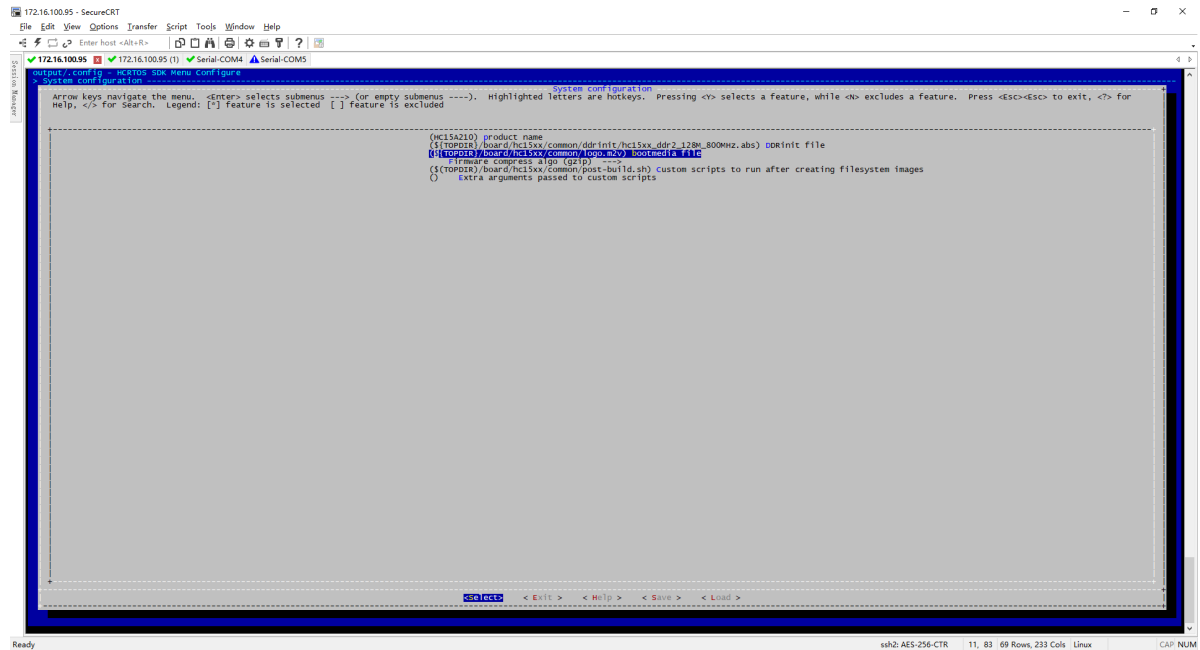
```
CONFIG_BOOT_SHOWLOGO=y
```

这样配置后，需要如下编译

```
make O=b1 hichip_hc15xx_db_b200_b1_defconfig
make O=b1 all
make hichip_hc15xx_db_a210_v12_hcdemo_defconfig
make all
```

12.3.1.2 如果是已经编译过的工程

1. 运行 `make menuconfig`，将 `system configuration` 中的 `bootmedia file` 按下图填写好：

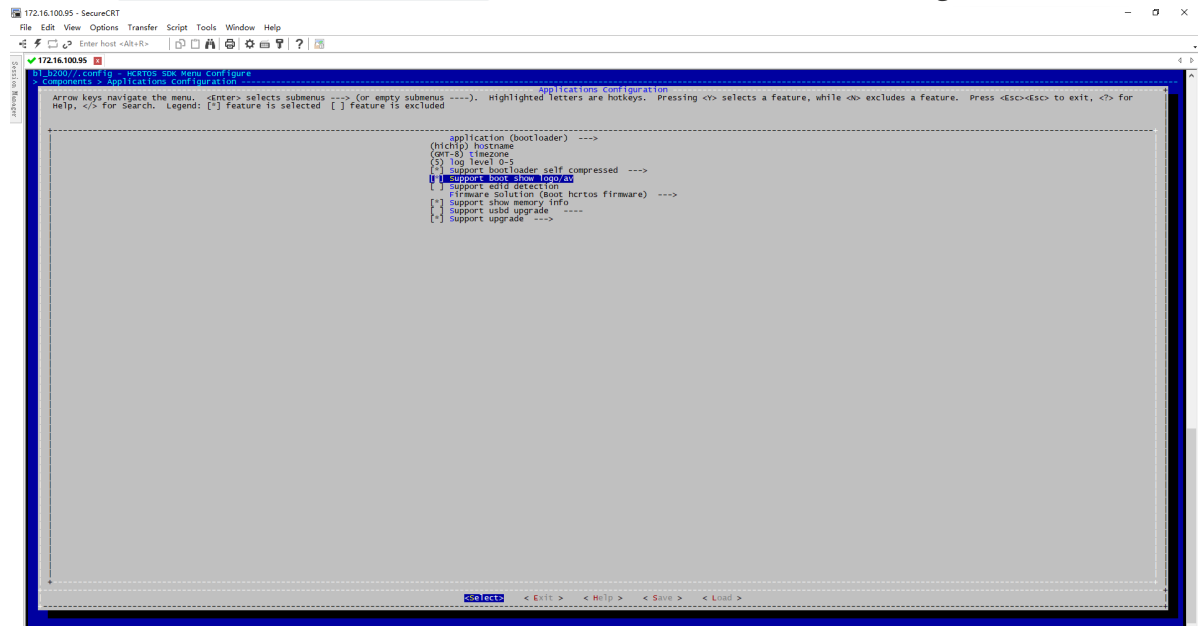


退出界面后保存：

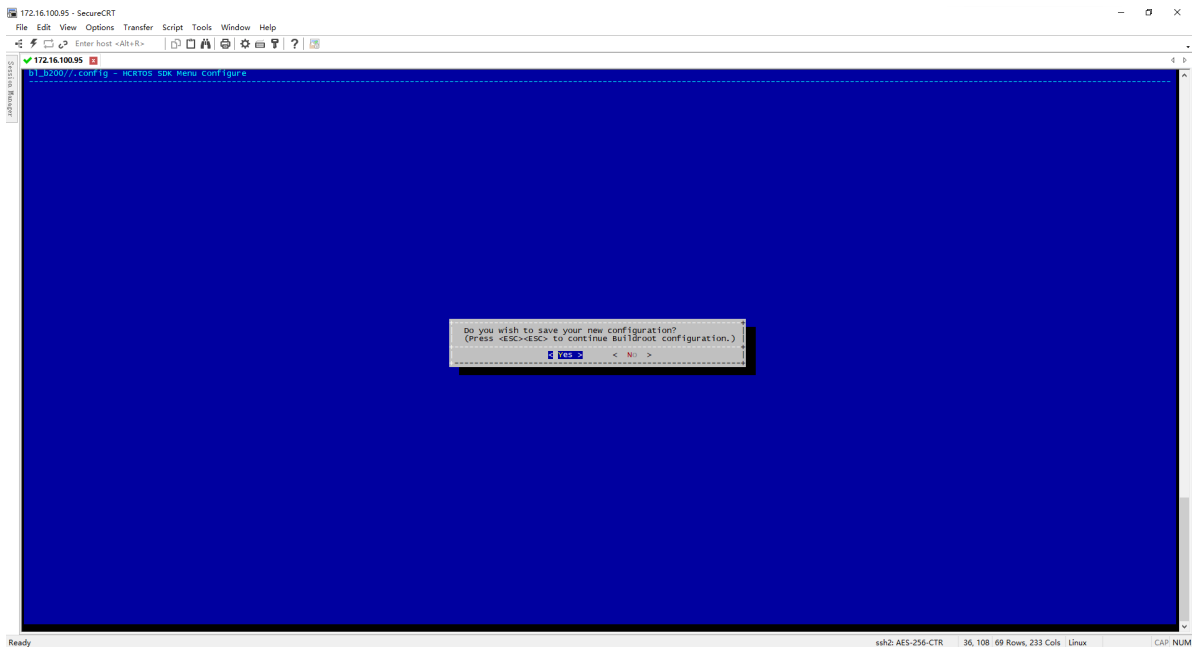
2. 进入 `bootloader` 的 `menuconfig` 进行配置：

```
make O=b1 menuconfig
```

进入如下 `Applications Configuration` 界面后，按空格键，打开 `boot show logo` 前的星号：



退出界面后保存：



3. 重新编译命令：

```
make O=b1 all
make all
```

12.3.2 如何关闭boot show logo

1. 参考13.3.1中开启boot show logo的配置，进入 `bootloader` 的 `menuconfig` 将 `support boot show logo/av` 前的星号取消。
2. 因为不需要show logo，所以可以将flash中的logo空间释放出来。在 `configs/hichip_hc15xx_db_a210_v12_hcdemo_defconfig` 中配置如下：

```
BR2_EXTERNAL_BOOTMEDIA_FILE=""
```

3. 减小flash中占用的空间，即减小 `hc15xx-db-a210.dts` 中 `romfs.img` 的大小：

```
sfspi {
    pinmux-active = <PINPAD_L11 1 PINPAD_L12 1 PINPAD_L13 1 PINPAD_L14
1>;

    sclk = <50000000>;
    dma-mode = <1>;
    status = "okay";

    spi_nor_flash {
        spi-tx-bus-width = <1>;
        spi-rx-bus-width = <1>;
        reg = <0>;
        status = "okay";
        partitions {
            part-num = <4>;

            /* part0 is for entire norflash by default */

            part1-label = "boot";
            part1-reg = <0x0 0x60000>;
            part1-filename = "bootloader.bin";

            part2-label = "eromfs";
```

```

        part2-reg = <0x60000 0x20000>;//0x60000为flash 地址, 0x20000
        为offset大小。两个相加即为下一个模块的0x80000
        part2-filename = "romfs.img";

        part3-label = "firmware";
        part3-reg = <0x80000 0x360000>;
        part3-filename = "hcdemo.uImage";

        part4-label = "persistentmem";
        part4-reg = <0x3e0000 0x20000>;
        part4-filename = "persistentmem.bin";
    };
};

```

如果修改了dts, 那么需要按如下方式进行编译:

```

make O=bl kernel-rebuild all
make kernel-rebuild all

```

12.3.3 如何更换boot show logo

1. 如果客户只是要换个logo, 那么直接更换logo.m2v即可。
2. 如果客户需要播放开机动画, 那么需要注意: 文件中只能有video, 不能有声音。格式必须是mpeg2格式。大小需要能放进flash空间。
3. 如何生成m2v: 可以到如下路径拿转换工具[ffmpeg-4.4-essentials build](https://gitlab.hichiptech.com:62443/sw/tools.git), 参照里面的命令介绍:
<https://gitlab.hichiptech.com:62443/sw/tools.git>

12.3.4 如何更换.h264的logo

1. 下载ffmpeg开源工具 https://www.gyan.dev/ffmpeg/builds/packages/ffmpeg-5.0.1-full_build.7z
2. 解压
3. (假设要把logo.bmp转换为264格式) 把logo.bmp copy到ffmpeg-5.0.1-full_build/bin 下
4. 执行windows cmd命令行, 并cd到ffmpeg-5.0.1-full_build/bin目录
5. 执行命令

```

ffmpeg -i "logo.bmp" -vf
scale=out_color_matrix=bt709:flags=full_chroma_int+accurate_rnd,format=yuv420p -
c:v libx264 -profile:v high -x264-params ref=1 -b:v 4000k -f rawvideo
logo.4000k.264

```

或者

```

ffmpeg -i "logo.bmp" -vf
scale=out_color_matrix=bt709:flags=full_chroma_int+accurate_rnd,format=yuv420p -
c:v libx264 -profile:v high -x264-params ref=1 -b:v 5000k -f rawvideo
logo.5000k.264

```

其中 -b:v 4000k 或者 -b:v 5000k 表示编码图像质量, 一般可以尝试3000k/4000k/5000k, 会有不同的图像质量, 对应的264格式的文件size也会不一样。用户需综合考虑flash空间以及图像质量进行选择。

12.4 屏幕如何做到旋转。

12.4.1 OSD的旋转

需要修改dts中的以下代码：

```
rotate {
    /*
     * anticlockwise,
     * support 0/90/180/270 degree.
     */
    rotate = <0>;

    /*
     * value is 0 or 1
     * h_flip = 1 is horizontally flipped
     * v_flip = 1 is vertically flipped
     */
    h_flip = <0>;
    v_flip = <0>;

    status = "disabled";/*modify to okay*/
};
```

12.4.2 视频的旋转：

通过调用以下函数可以实现0

```
int hcplayer_change_rotate_type(void *player, rotate_type_e rotate_type);
```

改完重新编译：

```
cd hcartos
make O=bl kernel-rebuild all
make kernel-rebuild all
```

12.4.3 一键旋转功能

目前在SDK中有一键旋转的例子：`\components\applications\apps-projector\source\src\projector\projector.c`

```
else if(KEY_ROTATE_DISPLAY == act_key || act_key == KEY_FLIP){
    set_next_flip_mode();
    projector_sys_param_save();
}
```

此处的一键旋转同时包括OSD的旋转和视频的旋转，具体可以参考 `set_next_flip_mode()` 的实现。

12.4.4 同屏投屏时如何旋转和设置分辨率

1. 设置旋转:

调用: `key_cast_rotate(is_active);`

实现:

```
static void key_cast_rotate(bool is_active)
{
    int rotate_type = ROTATE_TYPE_0;

    if (data_mgr_cast_rotation_get()){
        rotate_type = ROTATE_TYPE_0;
        data_mgr_cast_rotation_set(0);
        // fbdev_set_rotate(0, 0, 0);
    }
    else{
        rotate_type = ROTATE_TYPE_270;
        data_mgr_cast_rotation_set(1);
        // fbdev_set_rotate(90, 0, 0);
    }
    #if defined(SUPPORT_FFPLAYER) || defined(__linux__)
    //api_logo_reshow();
    void *player = api_ffmpeg_player_get();
    if(player !=NULL && !is_active){
        hcplayer_change_rotate_type(player, rotate_type);
        hcplayer_change_mirror_type(player, MIRROR_TYPE_NONE);
    }
    #endif
}
```

2. 设置分辨率

```
tv_sys_app_set(APP_TV_SYS_720P);
tv_sys_app_set(APP_TV_SYS_1080P);
```

12.5 SD卡驱动

首先需要打开menuconfig中的 `components -> prebuilts -> sd-mmc driver`.

修改对应的dts中的配置, 以下以B200 V2.2为例, 需修改为如下:

```
mmc {
    //hertos-compatible = "hichip,dw-mshc";
    // compatible = "snps,dw-mshc";
    reg = <0x1884C000 0x2000>;
    card-detect-delay = <200>;
    clock-frequency = <198000000>;
    interrupts = <10>;
    pinmux-active = <PINPAD_L16 4 PINPAD_L17 4 PINPAD_L18 4 PINPAD_L19 4
PINPAD_L20 4 PINPAD_L21 4>;//对应原理图上的pin脚
    // bus-width = <8>;
    bus-width = <4>;
}
```



```
// bus-width = <1>;
broken-cd;
cd-gpios = <PINPAD_B02 0>;//插拔检测脚
//non-removable;

cap-sd-highspeed;
sd-uhs-sdr12;
sd-uhs-sdr25;
sd-uhs-sdr50;
status = "okay";
};
```

12.6 LVGL最高支持的帧数配置

12.7 如何在demo配置之外，增加定制化的板级配置

在hichip freeRTOS SDK的 configs 目录中，有很多的defconfig，其命名的规则如下：

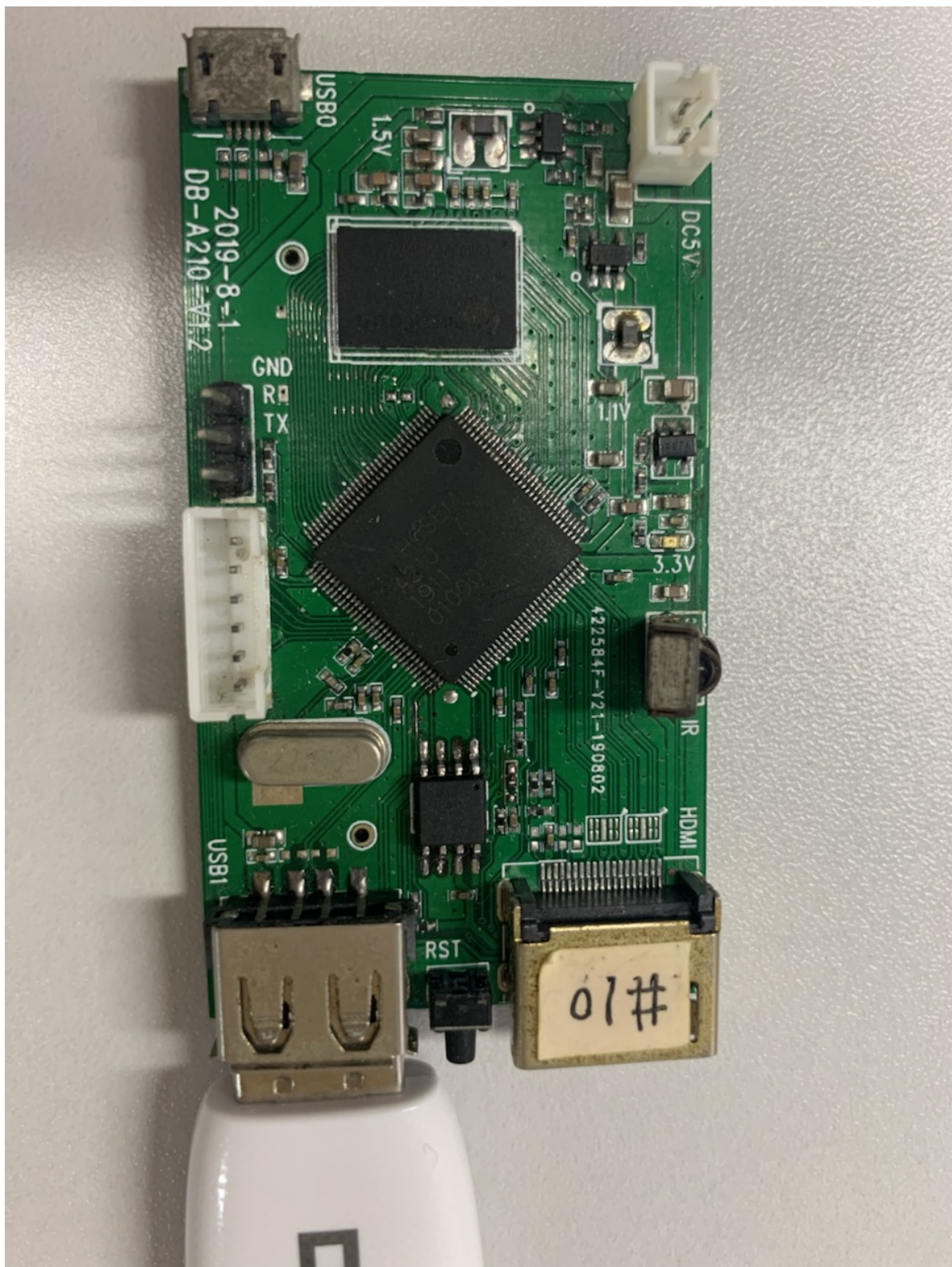
```
example:
hichip_hc16xx_db_a3100_v10_hcdemo_usbcast_bl_defconfig
hichip_hc16xx_db_a3100_v10_hcdemo_usbcast_defconfig
ex:
hichip_<chip_number>_<board_name>_<feature>_bl_defconfig
hichip_<chip_number>_<board_name>_<feature>_defconfig
```

关于chip_number:

1. hc15xx包括A110 / A210 / A211C / B100 / B200 / B210 等
2. hc16xx包括 A3100 / A3200 / A3300 / A5100 / B3100 / C3000 / C5200 / D3000 / D5200 / B300 / C300 等。

关于board_name:

基本可以在demo板子上看到，如下图即为db_a210_v12。



关于feature:

1. hcdemo: 对于选择的是hcdemo application, 支持本地多媒体 (如果板子有hdmi rx, cvbs in等, 则也支持)
2. hcdemo_usbcast : 本地多媒体 + 有线同屏
3. hcdemo_wificast : 本地多媒体 + 无线同屏
4. hcdemo_cast : 本地多媒体 + 有线同屏 + 无线同屏
5. hcscreen: 仅有线同屏 + 无线同屏

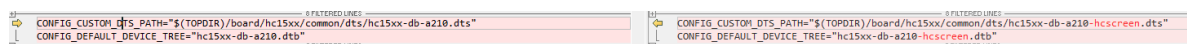
12.7.1 增加一个板级配置

如果需要增加配置，可以按如下步骤：

1. 在 configs 中增加两个配置，一个bl的配置，一个app的配置，例如：

```
hichip_hc15xx_db_a210_v12_hcdemo_bl_defconfig
hichip_hc15xx_db_a210_v12_hcdemo_defconfig
```






2. 修改 hichip_hc15xx_db_a210_v12_hcdemo_bl_defconfig 和 hichip_hc15xx_db_a210_v12_hcdemo_defconfig 中使用的dts配置文件。



3. 修改 hichip_hc15xx_db_a210_v12_hcdemo_defconfig 中使用的app feature，按12.7节中介绍的命名规则进行选择。



4. 在 \board\hc15xx\common\dts 中增加步骤2中，defconfig里改动的dts名字

| board > hc15xx > common > dts | | | | 在 dts 中搜索 | |
|---|------------------|----------------------|-------|-----------|--|
| 名称 | 修改日期 | 类型 | 大小 | | |
|  hc15xx-db-a110.dts | 2022/11/12 9:45 | DTS Audio File (...) | 10 KB | | |
|  hc15xx-db-a210.dts | 2022/11/12 9:45 | DTS Audio File (...) | 10 KB | | |
|  hc15xx-db-a210-hccast.dts | 2022/11/12 9:45 | DTS Audio File (...) | 10 KB | | |
|  hc15xx-db-a210-hcscreen.dts | 2022/11/12 9:45 | DTS Audio File (...) | 10 KB | | |
|  hc15xx-db-b100.dts | 2022/11/12 14:44 | DTS Audio File (...) | 14 KB | | |

5. 重新编译

```
make O=bl hichip_hc15xx_db_a210_v12_hcdemo_bl_defconfig
make O=bl all
make hichip_hc15xx_db_a210_v12_hcdemo_defconfig
make all
```

6. 如需修改配置，可以 make O=bl menuconfig 和 make menuconfig。如需修改dts，则编辑后需进行重新编译

```
make O=bl kernel-rebuild all
make kernel-rebuild all
```

12.8 hcRTOS Nor Flash区间的分布

12.8.1 流程介绍

1. 在post-build.sh中，会对每个partition进行打包，并生成最终烧录文件：

```
if [ -f ${IMAGES_DIR}/hcboot.out ] && [ -f ${IMAGES_DIR}/hcboot.bin ] && [ -f  
${BR2_EXTERNAL_BOARD_DDRINIT_FILE} ]; then  
    message "Generating bootloader.bin ....."  
    fddrinit=$(basename ${BR2_EXTERNAL_BOARD_DDRINIT_FILE})
```

```

hcboot_sz=$(wc -c ${IMAGES_DIR}/hcboot.bin | awk '{print $1}')
hcboot_ep=$(readelf -h ${IMAGES_DIR}/hcboot.out | grep Entry | awk '{print $NF}')

${DDRCONFIGMODIFY} --input ${BR2_EXTERNAL_BOARD_DDRINIT_FILE} --output
${IMAGES_DIR}/${fddrinit} \
    --size ${hcboot_sz} \
    --entry ${hcboot_ep} \
    --from 0xafc02000 \
    --to ${hcboot_ep}

cat ${IMAGES_DIR}/${fddrinit} ${IMAGES_DIR}/hcboot.bin >
${IMAGES_DIR}/bootloader.bin
message "Generating bootloader.bin done!"
fi
//以上为hcboot的在flash中的分布
if [ -f ${IMAGES_DIR}/${app}.bin ] ; then
    message "Generating ${app}.uImage ....."
    if [ "${BR2_EXTERNAL_FW_COMPRESS_LZO1X}" = "y" ] ; then

        ${HOST_DIR}/bin/hcprecomp2 ${IMAGES_DIR}/${app}.bin
        ${IMAGES_DIR}/${app}.bin.lzo
        ${HOST_DIR}/bin/mkimage -A mips -O u-boot -T standalone -C lzo -n ${app}
        -e ${app_ep} -a ${app_load} \
            -d ${IMAGES_DIR}/${app}.bin.lzo ${IMAGES_DIR}/${app}.uImage

    elif [ "${BR2_EXTERNAL_FW_COMPRESS_GZIP}" = "y" ] ; then

        gzip -kf9 ${IMAGES_DIR}/${app}.bin > ${IMAGES_DIR}/${app}.bin.gz
        ${HOST_DIR}/bin/mkimage -A mips -O u-boot -T standalone -C gzip -n
        ${app} -e ${app_ep} -a ${app_load} \
            -d ${IMAGES_DIR}/${app}.bin.gz ${IMAGES_DIR}/${app}.uImage

    elif [ "${BR2_EXTERNAL_FW_COMPRESS_LZMA}" = "y" ] ; then

        lzma -zkf -c ${IMAGES_DIR}/${app}.bin > ${IMAGES_DIR}/${app}.bin.lzma
        ${HOST_DIR}/bin/mkimage -A mips -O u-boot -T standalone -C lzma -n
        ${app} -e ${app_ep} -a ${app_load} \
            -d ${IMAGES_DIR}/${app}.bin.lzma ${IMAGES_DIR}/${app}.uImage

    fi
    message "Generating ${app}.uImage done!"
fi
//以上为SDK软件在flash中的分布
if [ -f ${BR2_EXTERNAL_BOOTMEDIA_FILE} ] ; then
    message "Generating logo.hc ....."
    ${GENBOOTMEDIA} -i ${BR2_EXTERNAL_BOOTMEDIA_FILE} -o ${IMAGES_DIR}/fs-
partition1-root/logo.hc
    message "Generating logo.hc done!"
    message "Generating romfs.bin ....."
    genromfs -f ${IMAGES_DIR}/romfs.img -d ${IMAGES_DIR}/fs-partition1-root/ -v
    "romfs"
    message "Generating romfs.bin done!"
fi
//以上为logo数据在flash中的分布
firmware_version=$(date +%y%m%d%H%M)

message "Generating persistentmem.bin ....."

```

```

tvtype=$(PATH=$PYPATH ${FDTINFO} --dtb ${DTB} --node /hcartos/de-engine --prop
tvtype)
volume=$(PATH=$PYPATH ${FDTINFO} --dtb ${DTB} --node /hcartos/i2so --prop volume)
${GENPERSISTENTMEM} -v ${firmware_version} -p ${BR2_EXTERNAL_PRODUCT_NAME} -V
${volume} -t ${tvtype} -o ${IMAGES_DIR}/persistentmem.bin
message "Generating persistentmem.bin done"
//以上为用户数据和系统数据等在flash中的分布

```

2. 在dts中定义每个partitions的位置和大小:

```

persistentmem {
    mtdname = "persistentmem";
    size = <4096>;
    status = "okay";
};

sfspi {
    pinmux-active = <PINPAD_L11 1 PINPAD_L12 1 PINPAD_L13 1 PINPAD_L14 1>;
    sclk = <500000000>;
    dma-mode = <1>;
    status = "okay";

    spi_nor_flash {
        spi-tx-bus-width = <1>;
        spi-rx-bus-width = <1>;
        reg = <0>;
        status = "okay";
        partitions {
            part-num = <4>;//flash中的part数，与下面的配置对应。

            /* part0 is for entire norflash by default */

            part1-label = "boot";
            part1-reg = <0x0 0x60000>;//起始0位置，长度0x60000
            part1-filename = "bootloader.bin";

            part2-label = "eromfs";
            part2-reg = <0x60000 0x20000>;
            part2-filename = "romfs.img";

            part3-label = "firmware";
            part3-reg = <0x80000 0x360000>;
            part3-filename = "hcdemo.uImage";

            part4-label = "persistentmem";
            part4-reg = <0x3e0000 0x20000>;
            part4-filename = "persistentmem.bin";
        };
    };
};

```

烧录到板子后，可以在命令行通过以下命令来查看

```

hc1512a@dba210# nsh //进入文件系统命令行
hc1512a@dba210(nsh)#
hc1512a@dba210(nsh)#
hc1512a@dba210(nsh)# ls dev/

```

```

/dev:
auddec
audsink
avsync0
avsync1
bus/
dis
dsc
fb0
ge
hdm1
i2c0
input/
mmz
mtd0
mtd1
mtd2
mtd3
mtd4
mtd5
mtdblock0          // for entire norflash by default
mtdblock1          // part1-filename
mtdblock2          // part2-filename
mtdblock3          // part3-filename
mtdblock4          // part4-filename
mtdblock5          // part5-filename
persistentmem
random
sndC0i2so
spidev0
uart1
uart_dummy
urandom
viddec
vidsink
hc1512a@dbA210(nsh)#

```

12.8.2 如何增加一个客户可读可写的分区？

在hichip freertos的norflash中，支持用户对nor flash的特定区域进行读写，方便客户存放定制化的数据。但需要按如下步骤进行：

1. 需要在dts中增加一个flash分区：

```

sfspi {
    pinmux-active = <PINPAD_L11 1 PINPAD_L12 1 PINPAD_L13 1 PINPAD_L14 1>;
    sclk = <50000000>;
    dma-mode = <1>;
    status = "okay";

    spi_nor_flash {
        spi-tx-bus-width = <1>;
        spi-rx-bus-width = <1>;
        reg = <0>;
        status = "okay";
        partitions {
            part-num = <5>;    // 增加后，这里的数量需要对应修改。

```



```

/* part0 is for entire norflash by default */

part1-label = "boot";
part1-reg = <0x0 0x19000>;
part1-filename = "bootloader.bin";

part2-label = "eromfs";
part2-reg = <0x19000 0x3C000>;
part2-filename = "romfs.img";

part3-label = "firmware";
part3-reg = <0x55000 0x38B000>;
part3-filename = "hcscreen.uImage";

part4-label = "user_data"; //增加的分区名
part4-reg = <0x3e0000 0x10000>; // 地址和大小。 0x10000表示分区
大小，必须与post-image.sh里的
part4-filename = "user_data.bin"; // 待烧录的文件。可以修改为客户文
件。

part5-label = "persistentmem";
part5-reg = <0x3f0000 0x10000>;
part5-filename = "persistentmem.bin";
};
};

```

2. 需要有如下修改：

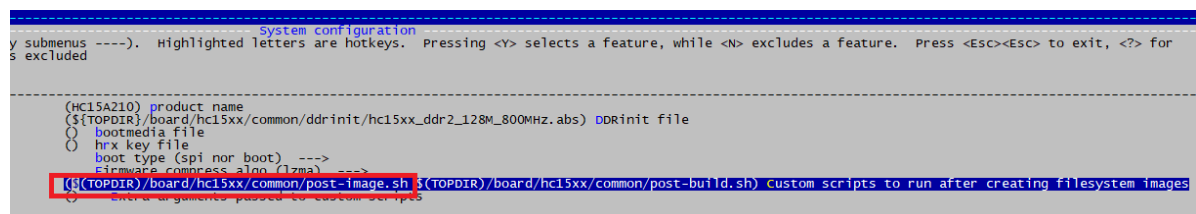
主要修改如下，代码编译阶段会通过post-image.sh在image目录下生成一个预置的user_data的文件夹，并格式化成user_data.bin文件。该文件就是放置于上述partition中的文件。

```

modified:    Makefile
new file:    board/hc15xx/common/dts/hc15xx-db-a210-hcscreen-p1.dts
new file:    board/hc15xx/common/post-image.sh
modified:    build/tools
new file:    configs/hichip_hc15xx_db_a210_hcscreen_p1_defconfig

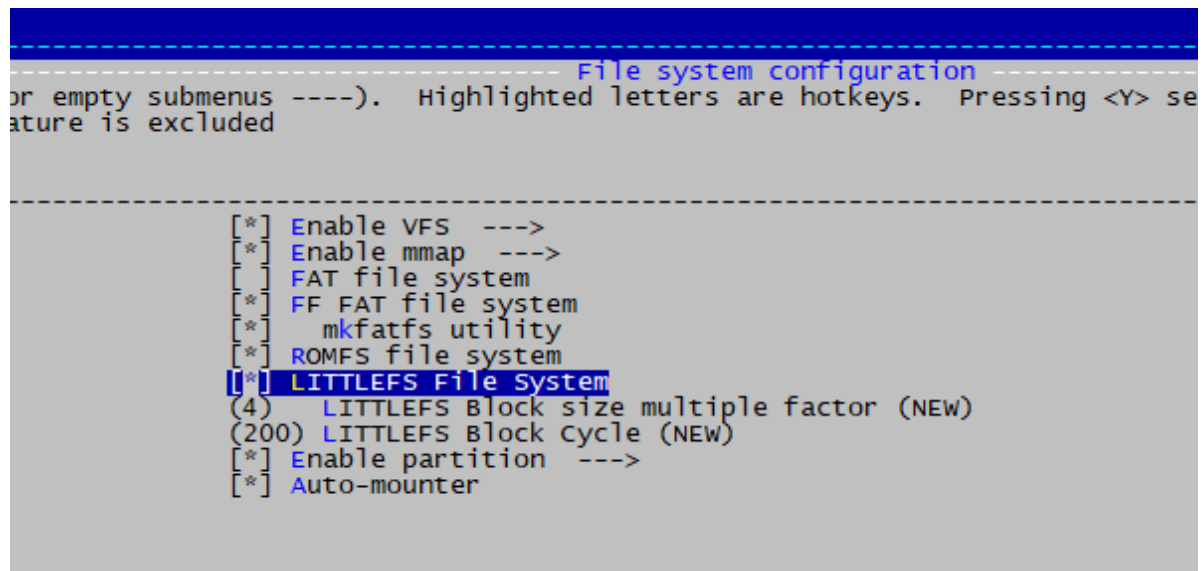
```

2.1 menuconfig中，需要增加进post-image.sh的改动：



2.2 menuconfig中，需要打开littlefs的文件系统配置：

Components > kernel > File system configuration > [*] LITTLEFS File System



2.3 post-image.sh中，-b 必须是4096，-s的大小需要与dts中的分区大小一致！！

```
1 #!/bin/bash
2
3 DTB=${IMAGES_DIR}/dtb.bin
4
5 if [ -d ${DATA_DIR} ] ; then
6     echo "Generating user_data.bin ...."
7     echo "1" > ${DATA_DIR}/.tmp
8
9     cp ${IMAGES_DIR}/dtb.bin ${IMAGES_DIR}/user_data/ -f
10    PYPATH=/usr/bin:/usr/local/bin:$PATH
11    PARTINFO=${TOPDIR}/build/scripts/partinfo.py
12    SIZE=$(PATH=$PYPATH ${PARTINFO} --dtb ${DTB} --partname user_data --get-size true)
13
14    echo "data partition size is ${SIZE}"
15    #mkfs.jffs2 -r ${DATA_DIR} -o ${IMAGES_DIR}/user_data.bin -e 0x4000 -pad ${SIZE} -n
16    ./build/tools/mklittlefs -c ${IMAGES_DIR}/user_data/ -d 5 -b 4096 -s 0x10000 ${IMAGES_DIR}/user_data.bin
17    echo "Generating user_data.bin done"
18 fi
```

2.4 在代码中每次启机都需要调用

```
ret = mount("/dev/mtdblock4", "/mnt/lfs", "littlefs", 0, NULL);
```

或者用串口命令：

```
mount -t littlefs /dev/mtdblock4 /mnt/lfs
```

该命令是把flash block mount成littlefs的文件系统，这样就可以调用fwrite等进行操作。

2.5 如果mount失败，需要调用：

```
ret = mount("/dev/mtdblock4", "/mnt/lfs", "littlefs", 0, "forceformat");
```

或者进入nsh命令行：

```
mount -t littlefs -o forceformat /dev/mtdblock4 /mnt/lfs
```

notice: 该调用可以把 user_data 分区 mount 到 /mnt/lfs 文件夹，并格式化成 littlefs 格式。这样里面的预置文件等都会丢失。如果分区里已经有了预置用户数据，那么一定不要用 forceformat 进行格式化。

3. 修改完后重新编译：

```
make O=bl kernel-rebuild all
make kernel-rebuild all
```

4. 修改完就可以通过标准 open/close 等文件操作函数进行开关/读写文件。读写代码可以参考如下：


```

char *filename = "/mnt/lfs/text.txt";
FILE *file = fopen(filename, "wt+");

if (!file) {
    printf("error fopen! \n");
}
int b[6] = {1,3,5,7,9,11};
int a[6] = {};
fwrite(b, sizeof(int), 6, file);
fflush(file);
fclose(file);

file = fopen(filename, "rt+");
ret = fread(a, sizeof(int), 6, file);
printf("ret = %d, %d, %d, %d, %d, %d, %d \n", ret, a[0], a[1], a[2],
a[3], a[4], a[5]);
fclose(file);
return ret;

```

12.8.3 如何临时添加一个不支持的flash

当你用了一个不支持的flash，会导致不能启机。会有如下打印：

```

ERROR: open /dev/mtdblock3
* kernel: default image load address = 0x00000000

```

这时，你可以使用wingdb直接download以下文件\output\images\xxxx.out到板子，其中xxxx是你选择的app的名字，比如，你用的hcscreen的app，那么就是hcscreen.out.

download完成并运行，你会得到如下打印：

```

D/HEX JEDEC id bytes:: 0000-0010: 0B 40 17 0B 40 17 .@..@.
D/HEX JEDEC id bytes:: 0000-0010: 0B 40 17 0B 40 17 .@..@.
D/HEX JEDEC id bytes:: 0000-0010: 0B 40 17 0B 40 17 .@..@.
D/HEX JEDEC id bytes:: 0000-0010: 0B 40 17 0B 40 17 .@..@.

```

改打印即为flash的ID，这样就可以结合flash的规格书，添加如下代码：

```
1 // SPDX-License-Identifier: GPL-2.0
2 /*
3  * This file is used to hold the spi-nor-manufacturer parts that datasheet unknown or to be verified.
4  *
5  * IMPORTANT: You'd better to ask help from spi-nor-manufacturer to confirm this.
6  */
7
8 #include <linux/mtd/spi-nor.h>
9
10 #include "core.h"
11
12 static const struct flash_info general_parts[] = {
13     { "25q32B", INFO(0x544016, 0, 64 * 1024, 64, SECT_4K ) },
14
15     // Zbit
16     { "25Q32B", INFO(0x5E4016, 0, 64 * 1024, 64, SECT_4K ) },
17
18     { "XT25F64F", INFO(0x0B4017, 0, 64 * 1024, 128, SECT_4K ) },
19 };
20
21 const struct spi_nor_manufacturer spi_nor_general = {
22     .name = "Unknow",
23     .parts = general_parts,
24     .nparts = ARRAY_SIZE(general_parts),
25 };
26
27 ~
28 ~
29 ~
30 ~
31 ~
32 ~
33 ~
34 ~
35 ~
36 ~
37 ~
38 ~
39 ~
40 ~
41 ~
42 ~
43 ~
44 ~
45 ~
46 ~
47 ~
48 ~
49 ~
50 ~
51 ~
52 ~
53 ~
54 ~
55 ~
56 ~
57 ~
58 ~
59 ~
60 ~
61 ~
62 ~
63 ~
64 ~
65 ~
66 ~
67 ~
68 ~
69 ~
70 ~
71 ~
72 ~
73 ~
74 ~
75 ~
76 ~
77 ~
78 ~
79 ~
80 ~
81 ~
82 ~
83 ~
84 ~
85 ~
86 ~
87 ~
88 ~
89 ~
90 ~
91 ~
92 ~
93 ~
94 ~
95 ~
96 ~
97 ~
98 ~
99 ~
100 ~
101 ~
102 ~
103 ~
104 ~
105 ~
106 ~
107 ~
108 ~
109 ~
110 ~
111 ~
112 ~
113 ~
114 ~
115 ~
116 ~
117 ~
118 ~
119 ~
120 ~
121 ~
122 ~
123 ~
124 ~
125 ~
126 ~
127 ~
128 ~
129 ~
130 ~
131 ~
132 ~
133 ~
134 ~
135 ~
136 ~
137 ~
138 ~
139 ~
140 ~
141 ~
142 ~
143 ~
144 ~
145 ~
146 ~
147 ~
148 ~
149 ~
150 ~
151 ~
152 ~
153 ~
154 ~
155 ~
156 ~
157 ~
158 ~
159 ~
160 ~
161 ~
162 ~
163 ~
164 ~
165 ~
166 ~
167 ~
168 ~
169 ~
170 ~
171 ~
172 ~
173 ~
174 ~
175 ~
176 ~
177 ~
178 ~
179 ~
180 ~
181 ~
182 ~
183 ~
184 ~
185 ~
186 ~
187 ~
188 ~
189 ~
190 ~
191 ~
192 ~
193 ~
194 ~
195 ~
196 ~
197 ~
198 ~
199 ~
200 ~
201 ~
202 ~
203 ~
204 ~
205 ~
206 ~
207 ~
208 ~
209 ~
210 ~
211 ~
212 ~
213 ~
214 ~
215 ~
216 ~
217 ~
218 ~
219 ~
220 ~
221 ~
222 ~
223 ~
224 ~
225 ~
226 ~
227 ~
228 ~
229 ~
230 ~
231 ~
232 ~
233 ~
234 ~
235 ~
236 ~
237 ~
238 ~
239 ~
240 ~
241 ~
242 ~
243 ~
244 ~
245 ~
246 ~
247 ~
248 ~
249 ~
250 ~
251 ~
252 ~
253 ~
254 ~
255 ~
256 ~
257 ~
258 ~
259 ~
260 ~
261 ~
262 ~
263 ~
264 ~
265 ~
266 ~
267 ~
268 ~
269 ~
270 ~
271 ~
272 ~
273 ~
274 ~
275 ~
276 ~
277 ~
278 ~
279 ~
280 ~
281 ~
282 ~
283 ~
284 ~
285 ~
286 ~
287 ~
288 ~
289 ~
290 ~
291 ~
292 ~
293 ~
294 ~
295 ~
296 ~
297 ~
298 ~
299 ~
300 ~
301 ~
302 ~
303 ~
304 ~
305 ~
306 ~
307 ~
308 ~
309 ~
310 ~
311 ~
312 ~
313 ~
314 ~
315 ~
316 ~
317 ~
318 ~
319 ~
320 ~
321 ~
322 ~
323 ~
324 ~
325 ~
326 ~
327 ~
328 ~
329 ~
330 ~
331 ~
332 ~
333 ~
334 ~
335 ~
336 ~
337 ~
338 ~
339 ~
340 ~
341 ~
342 ~
343 ~
344 ~
345 ~
346 ~
347 ~
348 ~
349 ~
350 ~
351 ~
352 ~
353 ~
354 ~
355 ~
356 ~
357 ~
358 ~
359 ~
360 ~
361 ~
362 ~
363 ~
364 ~
365 ~
366 ~
367 ~
368 ~
369 ~
370 ~
371 ~
372 ~
373 ~
374 ~
375 ~
376 ~
377 ~
378 ~
379 ~
380 ~
381 ~
382 ~
383 ~
384 ~
385 ~
386 ~
387 ~
388 ~
389 ~
390 ~
391 ~
392 ~
393 ~
394 ~
395 ~
396 ~
397 ~
398 ~
399 ~
400 ~
401 ~
402 ~
403 ~
404 ~
405 ~
406 ~
407 ~
408 ~
409 ~
410 ~
411 ~
412 ~
413 ~
414 ~
415 ~
416 ~
417 ~
418 ~
419 ~
420 ~
421 ~
422 ~
423 ~
424 ~
425 ~
426 ~
427 ~
428 ~
429 ~
430 ~
431 ~
432 ~
433 ~
434 ~
435 ~
436 ~
437 ~
438 ~
439 ~
440 ~
441 ~
442 ~
443 ~
444 ~
445 ~
446 ~
447 ~
448 ~
449 ~
450 ~
451 ~
452 ~
453 ~
454 ~
455 ~
456 ~
457 ~
458 ~
459 ~
460 ~
461 ~
462 ~
463 ~
464 ~
465 ~
466 ~
467 ~
468 ~
469 ~
470 ~
471 ~
472 ~
473 ~
474 ~
475 ~
476 ~
477 ~
478 ~
479 ~
480 ~
481 ~
482 ~
483 ~
484 ~
485 ~
486 ~
487 ~
488 ~
489 ~
490 ~
491 ~
492 ~
493 ~
494 ~
495 ~
496 ~
497 ~
498 ~
499 ~
500 ~
501 ~
502 ~
503 ~
504 ~
505 ~
506 ~
507 ~
508 ~
509 ~
510 ~
511 ~
512 ~
513 ~
514 ~
515 ~
516 ~
517 ~
518 ~
519 ~
520 ~
521 ~
522 ~
523 ~
524 ~
525 ~
526 ~
527 ~
528 ~
529 ~
530 ~
531 ~
532 ~
533 ~
534 ~
535 ~
536 ~
537 ~
538 ~
539 ~
540 ~
541 ~
542 ~
543 ~
544 ~
545 ~
546 ~
547 ~
548 ~
549 ~
550 ~
551 ~
552 ~
553 ~
554 ~
555 ~
556 ~
557 ~
558 ~
559 ~
560 ~
561 ~
562 ~
563 ~
564 ~
565 ~
566 ~
567 ~
568 ~
569 ~
570 ~
571 ~
572 ~
573 ~
574 ~
575 ~
576 ~
577 ~
578 ~
579 ~
580 ~
581 ~
582 ~
583 ~
584 ~
585 ~
586 ~
587 ~
588 ~
589 ~
590 ~
591 ~
592 ~
593 ~
594 ~
595 ~
596 ~
597 ~
598 ~
599 ~
600 ~
601 ~
602 ~
603 ~
604 ~
605 ~
606 ~
607 ~
608 ~
609 ~
610 ~
611 ~
612 ~
613 ~
614 ~
615 ~
616 ~
617 ~
618 ~
619 ~
620 ~
621 ~
622 ~
623 ~
624 ~
625 ~
626 ~
627 ~
628 ~
629 ~
630 ~
631 ~
632 ~
633 ~
634 ~
635 ~
636 ~
637 ~
638 ~
639 ~
640 ~
641 ~
642 ~
643 ~
644 ~
645 ~
646 ~
647 ~
648 ~
649 ~
650 ~
651 ~
652 ~
653 ~
654 ~
655 ~
656 ~
657 ~
658 ~
659 ~
660 ~
661 ~
662 ~
663 ~
664 ~
665 ~
666 ~
667 ~
668 ~
669 ~
670 ~
671 ~
672 ~
673 ~
674 ~
675 ~
676 ~
677 ~
678 ~
679 ~
680 ~
681 ~
682 ~
683 ~
684 ~
685 ~
686 ~
687 ~
688 ~
689 ~
690 ~
691 ~
692 ~
693 ~
694 ~
695 ~
696 ~
697 ~
698 ~
699 ~
700 ~
701 ~
702 ~
703 ~
704 ~
705 ~
706 ~
707 ~
708 ~
709 ~
710 ~
711 ~
712 ~
713 ~
714 ~
715 ~
716 ~
717 ~
718 ~
719 ~
720 ~
721 ~
722 ~
723 ~
724 ~
725 ~
726 ~
727 ~
728 ~
729 ~
730 ~
731 ~
732 ~
733 ~
734 ~
735 ~
736 ~
737 ~
738 ~
739 ~
740 ~
741 ~
742 ~
743 ~
744 ~
745 ~
746 ~
747 ~
748 ~
749 ~
750 ~
751 ~
752 ~
753 ~
754 ~
755 ~
756 ~
757 ~
758 ~
759 ~
760 ~
761 ~
762 ~
763 ~
764 ~
765 ~
766 ~
767 ~
768 ~
769 ~
770 ~
771 ~
772 ~
773 ~
774 ~
775 ~
776 ~
777 ~
778 ~
779 ~
780 ~
781 ~
782 ~
783 ~
784 ~
785 ~
786 ~
787 ~
788 ~
789 ~
790 ~
791 ~
7
```

添加完成后，编译：

```
make O=bl kernel-rebuild all
make kernel-rebuild all
```

12.9 uart蓝牙的配置

12.9.1 串口蓝牙

一般就两种模式：

1. 数据模式
2. 音频模式

一般音频模式下，音频数据直接通过蓝牙芯片，传到speaker中进行播放。

在数据模式下，蓝牙芯片会通过串口将数据透传到海奇芯片。需要配置如下：

dt部分

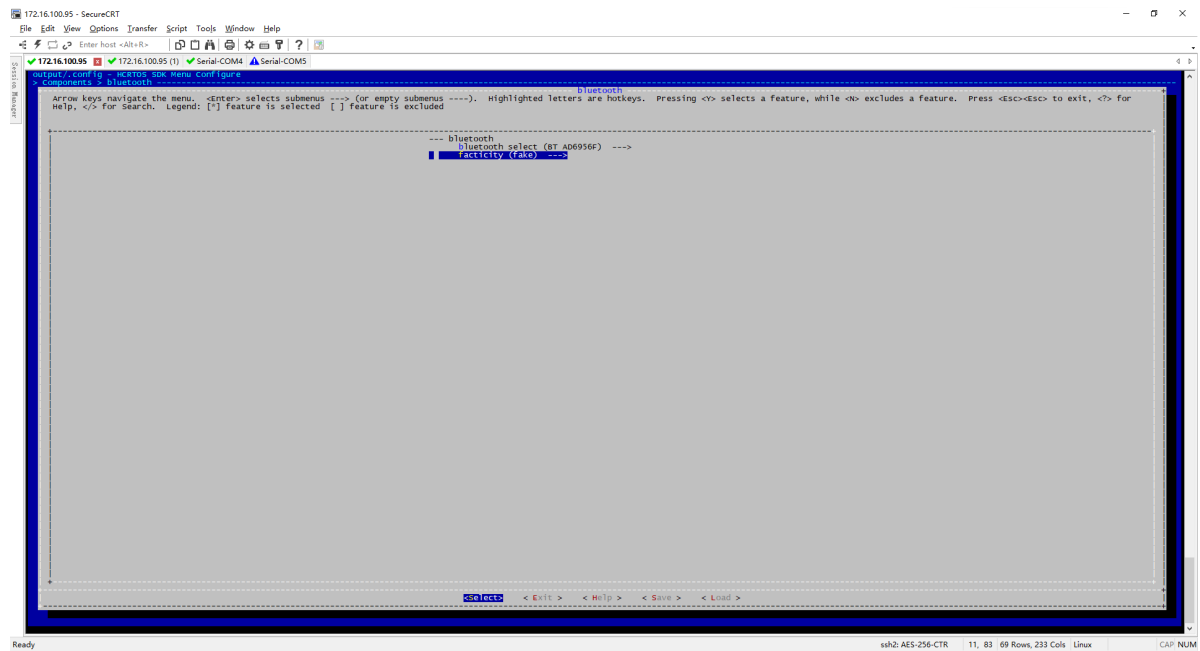
```
bluetooth {
    devpath = "/dev/uart1"; // 蓝牙硬件所连接的串口
    status = "disabled"; // 使能
};
```

如果只做透传，那么以上配置即可。客制化app可以对透传数据进行解析处理。

如果需要数据交互，可以按如下进行配置：

menuconfig部分，进入 Components > bluetooth进行蓝牙驱动选型。

该驱动对蓝牙原厂spec的数据传输进行了一些封装：



12.10 wifi的支持

目前hichip freeRTOS已经支持的wifi型号有：RTL8188FU、RTL8188EU、RTL8811CU。

```
finn.wei@hichip01:b100_mira_release$ tree components/hcilib/rtl8*
components/hcilib/rtl8188eu          //wifi 驱动，目前以库的形式提供
├─ Kconfig
└─ rtl8188eu.mk
components/hcilib/rtl8188fu          //wifi 驱动，目前以库的形式提供
├─ Kconfig
└─ rtl8188fu.mk
components/hcilib/rtl8811cu          //wifi 驱动，目前以库的形式提供
├─ Kconfig
└─ rtl8811cu.mk
components/hcilib/wpa_supplicant
├─ Config.in
├─ wpa_supplicant
├─ wpa_supplicant.mk
└─ wpa_supplicant-rtl              //realtek wpa_supplicant, , 目前以库的形式提供
```

手动配网可以参考 hcrtos_wifi_user_guide.pdf

12.11 TP驱动的集成和测试

驱动代码如下：\components\kernel\source\drivers\input\tp

测试代码：\components\cmds\source\input_event

以hy46xx为例，需要的配置如下：

1. 打开驱动

```

output/.config - HCRITOS SDK Menu Configure
Components > kernel > Drivers > input event > tp menu

There is no help available for this option.
Symbol: CONFIG_HC_HY46XX [=y]
Type : bool
Prompt: hy46xx
Location:
  -> Components
    -> kernel (BR2_PACKAGE_KERNEL [=y])
      -> Drivers
        -> input event (CONFIG_DRV_INPUT [=y])
          -> tp menu (CONFIG_TP [=y])
Defined at tp:6
Depends on: BR2_PACKAGE_KERNEL [=y] && CONFIG_DRV_INPUT [=y] && CONFIG_TP [=y]

```

2. 配置pin脚，需要配置一个I2C节点，再配置挂在I2C上的设备节点。

```

},

i2c@0{
    pinmux-active = <PINPAD_L28 3 PINPAD_L29 3>;
    device_type = "hichip,hcrtos-setup-setbit";
    reg_bit = <0xb8800094 18 1>;
    devpath = "/dev/i2c0";
    baudrate = <1000000>;
    mode = "master";
    status = "okay";
};

hy46xx_ts{
    i2c-devpath = "/dev/i2c0";
    i2c-addr = <0x38>;
    reset-gpio = <PINPAD_L24 0>;
    irq-gpio = <PINPAD_L27 0>;
    status = "okay";
};

```

3. 测试代码的打开：

```

output/.config - HCRITOS SDK Menu Configure
Components  Cmds
Cmds
There is no help available for this option.
Symbol: CONFIG_CMDS_INPUT [=n]
Type : bool
Prompt: input event operations
Location:
  -> Components
    -> Cmds (BR2_PACKAGE_CMDS [=y])
Defined at source:51
Depends on: BR2_PACKAGE_CMDS [=y] && CONFIG_DRV_INPUT [=y]
-- Cmds ilt subdir

```

3. 配置完成后重新编译：

```

make O=bl kernel-rebuild all
make kernel-rebuild hc-examples-rebuild cmds-rebuild all

```

12.12 hdmi in调试

下面以a3100方案为例，调试hdmi in。

1.涉及的文件

1. 驱动

/components/prebuilts/sysroot/usr/lib/libviddrv_hdmirx.a //hdmi rx驱动库文件
components/kernel/source/include/uapi/hcuapi/hdmi_rx.h //hdmi rx
驱动头文件

2. 配置文件

board\hichip\hc16xx\dts\hc16xx-db-a3100-v10-avp.dtsi
board\hichip\hc16xx\dts\hc16xx-db-a3100-v10.dtsi
configs\hichip_hc16xx_linux_avp_defconfig

3. demo代码

components/hc-examples/source/hdmi_rx_test.c //hdmi rx 测试程序
components/hc-examples/source/hdmi_switch //hdmi tx/rx demo程序

2. 配置hdmi in

(1). dts中使能hdmi rx

修改hc16xx-db-a3100-v10-avp.dtsi, 使能hdmi rx,

hdmi rx显示到DE和OSD, 配置是不同的, 请事先确认输出方式

ioctl(hdmi_rx_fd, HDMI_RX_SET_VIDEO_DATA_PATH, HDMI_RX_VIDEO_TO_DE); //DE显示

ioctl(hdmi_rx_fd, HDMI_RX_SET_VIDEO_DATA_PATH, HDMI_RX_VIDEO_TO_OSD); //OSD显示

```
#define CONFIG_HDMI_RX_SUPPORT 1 //置1, 使能hdmi rx

//de和osd配置的buffer大小不一样
#define CONFIG_MM_VIN_FB_SIZE (1920*1088*2*4) /*HDMI RX:OSD:1920*1088*2*2*4
DE:1920*1088*2*4*/

//sdk默认没有配置fb0, 请配置上
fb0 {
    bits_per_pixel = <32>;
    xres = <1280>;
    yres = <720>;
    xres_virtual = <1280>;
    yres_virtual = <720>;
    xoffset = <0>;
    yoffset = <0>;

    scale = <1280 720 1920 1080>;

    reg = <CONFIG_FB0_REG 0x1000>;
    /*
     * frame buffer memory from:
     * system : malloc buffer from system heap
     * mmz0 : malloc buffer from mmz0
     * none : set buffer by user
     * default is system if the property is missing
     */
    buffer-source = "none"; //hdmi in模式下, OSD模式必须填写none。 DE模式就还是
system, 不要改。
    status = "okay";
};

i2s {
```

```
pinmux-clock = <2 2 3 2 4 2>;
status = "okay";           //确保i2s使能
ejtag = "disabled";
};

//如果需要录制hdmi rx的声音, 需要将i2si enable
i2si {
    volume = <255>;
    status = "okay";
};
```

修改hc16xx-db-a3300-v10.dtsi, 将fb0配置到硬件FB1上。

```
&fb0 {
    .....
    reg = <CONFIG_FB1_REG 0x1000>; //配置成FB1
    .....
};
```

(2). 配置defconfig

(2-1). 进行menuconfig配置

```
make menuconfig
```

选上hdmi rx driver plugin

```
Components---> prebuilts---> hdmi rx driver
```

选上hdmi rx driver

```
Components---> kernel---> Drivers---> Video Support---> hdmi rx Support
```

选上fb

```
Components---> kernel---> Drivers---> video---> hcfb
```

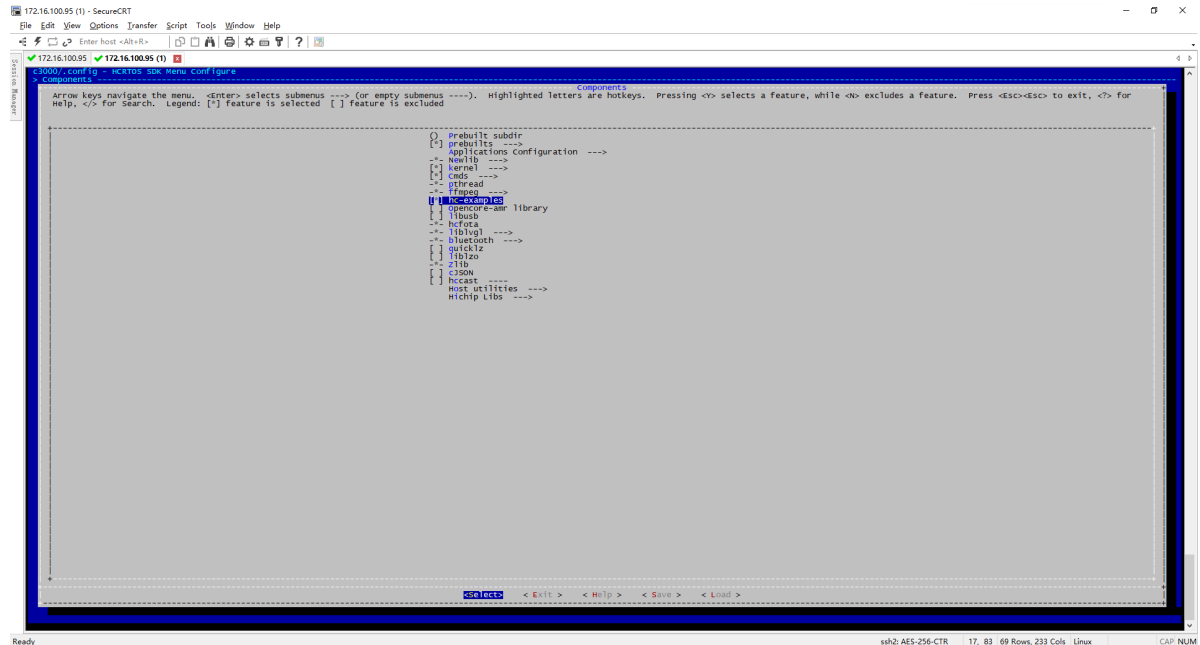
退出并保存配置


```
make menuconfig
```

使能hdmi_switch和hdmi rx test, 因为他们都在hc_examples里面, 选上后, 再进入cmds里选上hdmi rx test operation.

```
Components---> hc-examples
```

```
Components---> hc-examples ---> cmds ---> hdmi rx test operation
```



3.编译hdmi in

编译之前, 退出并保存好配置

```
make O=b1 kernel-rebuild all
make kernel-rebuild hc-examples-rebuild cmds-rebuild all
```

4.debug测试

(1). hdmi_switch

在命令行中输入: hdmi_switch

(2). hdmi rx

在命令行中输入hdmi_rx, 进入测试模式。

图像输出到TV, 命令: start -a 0 -v 2 //声音输出到hdmi tx, 图像输出到OSD

图像输出到LCD, 命令: start -a 1 -v 2 //声音输出到hdmi i2so, 图像输出到OSD

录制hdmi rx数据, 命令: start -a 3 -v 3 //声音和视频录制到U盘中

5.问题排查方法

情况1.hdmi设备有插入, 但是没有图像出来

问题现象: hdmi设备有插入, 但是没有图像出来

串口打印:

```
#tail -f /dev/virtuart & //打开后台打印
```



```
#hdmi_rx
hdmi_rx:# start -a 0 -v 2
apath 0, vpath 2
hdmi_rx start ok```
hdmi_rx:# hdmi_rx_enc_enable = 0
hdmi_rx_enc_play_enable = 0
hdmi_rx_enc_store_enable = 0
vin_param.fb_size = 0x17e8000
vin_param.fb_addr = 0xa5d67000
The framebuffer device was opened successfully.
variable screen: 1280x720, 32bpp
The framebuffer device was mapped to memory successfully.
hdmi_rx_open 422
HDMI_RX_FLAG_HPD_CONNECT
hdmi_rx_cb_video_clk_pre_chg_for_osd
exit hdmi_rx_cb_video_clk_pre_chg_for_osd
hdmi_rx_phy_init done
hdmi_rx_phy_wait_clock_lock TIME OUT
disconnect
hdmi_rx_ctrl_hal_enable_md_interrupt
```

问题原因：hdmi设备被识别，但是没有收到图像

1. hdmi设备本身就没有出图像
2. hdmi设备和D3100 hdmi rx存在兼容性问题，造成 hdmi设备不输出设备

排查方法：将hdmi设备查到TV，看看是否有图像输出。

1. 如果没有图像输出，那就是问题原因1，需要对hdmi设备重新上电。
2. 如果有图像输出，那就是问题原因2，是hdmi rx兼容性问题，需要联系原厂支持。

12.13 OSD 图层介绍和分辨率的修改

hichip freeRTOS支持双层OSD buffer，fb0和fb1，目前demo程序的OSD均显示在fb0上，性能足够（ddr size）的情况下，客制化程序可以同时使用fb0和fb1，使用方法一样。

示例参考代码：

```
components/hc-examples/source/fb_dither_test.c
components/hc-examples/source/fb_test.c
```

如果需要修改OSD的宽高，那么可以按下面的说明修改dts配置。

```
fb0 {
    bits_per_pixel = <16>; // BPP，支持16bpp / 32
    bpp
    xres = <1280>; // H
    yres = <720>; // V
    xres_virtual = <1280>; // H total buffer
    yres_virtual = <720>; // V total buffer
    xoffset = <0>; // display H offset
    yoffset = <0>; // display V offset
```

```

    scale = <1280 720 1920 1080>;                                //<xres, yres,
scale_xres(driver get from screen, not use), scale_yres>

    reg = <CONFIG_FB0_REG 0x1000>;                                //fb0 寄存器
/*
 * frame buffer memory from:
 * system : malloc buffer from system heap
 * mmz0    : malloc buffer from mmz0
 * none    : set buffer by user
 * default is system if the property is missing
 * extra-buffer-size = <0x9CA000>;
 */
    buffer-source = "system";                                     // buffer from system heap
memory
    extra-buffer-size = <0x3b9000>;                               // size = xres * yres * bpp
/ 4 * 3 + 128 * 1024
    // 其中*3表示 底层使用双buffer + 底层自用一个buffer, 128* 1024为lvg1自用调用栈, 需
    根据H/V进行调整。
    //extra-buffer-size = <0x86400>;
    status = "okay";                                             // okay is enable, the
other is disabled
};

fb1 {
    reg = <CONFIG_FB1_REG 0x1000>;                                // fb1 is not used in hichip
demo.
    status = "disabled";
};

```

修改完后重新编译:

```

make O=b1 kernel-rebuild all
make kernel-rebuild all

```

12.14 VIDEO图层的介绍和使用、测试

hichip freeRTOS 视频目前支持主图层和辅图层两层, 分别为DIS_LAYER_MAIN和DIS_LAYER_AUXP, 视频播放目前都在主图层。

测试代码:

components/hc-examples/source/dis_test.c

components/hc-examples/source/dis_test.h

命令格式可以在以下文件看到: components/hc-examples/source/ffplayer_examples.c

头文件: components/kernel/source/include/uapi/hcuapi/dis.h

12.15 按键类支持

12.15.1 如何增加ir遥控器

1.涉及的文件

1. 驱动代码

```
components/kernel/source/drivers/input
├─ input.c
├─ input-mt.c
├─ kconfig
├─ Makefile
├─ rc
│   ├─ hc_rc.c           // ir platform driver
│   ├─ ir-nec-decoder.c
│   ├─ keymaps
│   │   ├─ kconfig
│   │   ├─ Makefile
│   │   ├─ rc-hcdemo.c    // 遥控器驱动，负责按键键值映射
│   │   └─ rc-projector-cl.c
│   └─ Makefile
├─ rc-core.h
├─ rc-core-priv.h
├─ rc-ir-raw.c
├─ rc-main.c
└─ rc-map.h
```

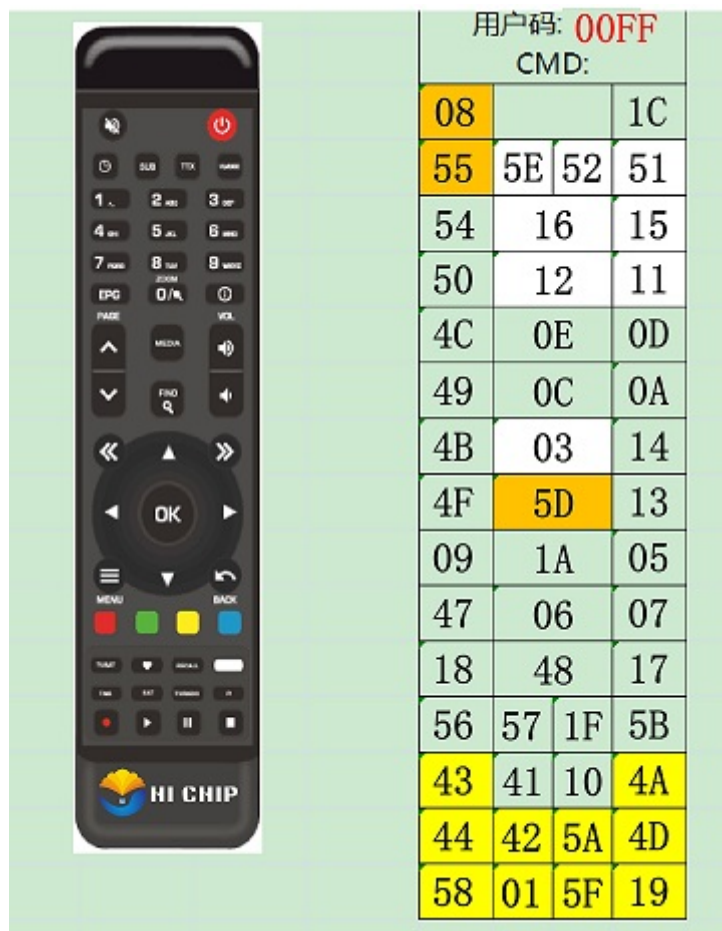
2.dts配置

```
board/hc16xx/common/dts/hc16xx-db-d3100-v20.dts
&irc {
    linux,rc-map-name = "rc-hcdemo";    // 对应rc_map_list中name，见rc-hcdemo.c中
    hcdemo_map数组定义
    pinctrl-names = "active";
    pinctrl-0 = <&pctl_irc>;
    status = "okay";
};
```

2.准备工作

拿到遥控器产品规格书，至少弄清楚如下两件事情：

- (1).遥控器协议，例如：NEC、RC-5、JVC等
- (2).遥控器按键码值表，如下图：



3.确认遥控器按键码值

通过input_test.c可以看到驱动发送的码值。

1.如果有遥控器键码表，请检查input_test.c里获取的键值是否与码值表一致

方法：进入menuconfig，勾选上：

Components > Cmds > [*] input event operations

保存并退出后，运行编译命令：make cmds-rebuild all

2.如果没有键码表，那么将遥控器每一个按键按一下，并且记录获得的对应码值

```
# input -i0 //进入测试命令，此时可以进行遥控按键

type:4, code:4, value:57094 // value 即为十进制的码值， 可以转换成16进制后，填入rc-
hcdemo.c或者rc-projector-c1.c中
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:57094
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:85
type:1, code:372, value:1
key 372 Pressed
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:85
type:0, code:0, value:0
( 0 0)
```

```
type:1, code:372, value:0
key 372 Released
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:57094
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:57094
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:57094
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:57094
type:0, code:0, value:0
( 0 0)
type:4, code:4, value:81
type:1, code:359, value:1
```

5.制作遥控器码值与按键映射表

1.freertos按键定义位于：`components/kernel/source/drivers/input/rc/keymaps/rc-hcdemo.c`
或者 `components/kernel/source/drivers/input/rc/keymaps/rc-projector-c1.c` 中

2.制作映射表，参考rc-hcdemo.c中struct rc_map_table hcdemo 填写映射表，上述遥控器的映射表如下：

```
static struct rc_map_table hcdemo[] = {
    .....
    { 0x54, KEY_NUMERIC_1 },
    { 0x16, KEY_NUMERIC_2 },
    { 0x15, KEY_NUMERIC_3 },
    { 0x50, KEY_NUMERIC_4 },
    { 0x12, KEY_NUMERIC_5 },
    { 0x11, KEY_NUMERIC_6 },
    { 0x4c, KEY_NUMERIC_7 },
    { 0x0e, KEY_NUMERIC_8 },
    { 0x0d, KEY_NUMERIC_9 },
    .....

    { 0x47, KEY_LEFT },
    { 0x1a, KEY_UP },
    { 0x07, KEY_RIGHT },
    { 0x48, KEY_DOWN },
    { 0x06, KEY_OK },
    .....
};
```

6.将映射表添加到rc-hcdemo驱动中

将映射表添加到rc-hcdemo.c文件的hcdemo[]数组中。

7.同时支持多款遥控器

- 1.按照“5.制作遥控器码值与按键映射表”，给每个遥控器制定映射表
- 2.将多个表合成一个映射表，如果码值有重复并且功能有冲突，那么说明某款遥控器无法同时支持
- 3.将“6.将映射表添加到rc-hcdemo驱动中”，更新hcdemo驱动

至此新遥控器支持已经完成，重新编译内核(`make kernel-rebuild all`)，烧写固件即可。

8.app如何响应按键

可以参考 `/components/applications/apps-hccast/source/src/hccast_app/key.c` 中代码

12.15.2 如何添加adc key支持

1.涉及的文件

```
1. 驱动代码
components/kernel/source/drivers/saradc
├─ adc_15xx_reg_struct.h
├─ adc_16xx_reg_struct.h
├─ hc_15xx_key_adc.c
├─ hc_15xx_key_adc.h
├─ hc_16xx_key_adc.c
├─ hc_16xx_key_adc.h
├─ hc_key_adc.c
├─ hc_saradc_clk_init.c
├─ hc_touch_adc.c
├─ hc_touch_adc.h
├─ kconfig
└─ Makefile

2. dts配置
board/hc16xx/common/dts/hc16xx-db-d3100-v20.dts
```

2.准备工作

| 功能键 | 电路图中电压(mv) | voltage_min | voltage_max | keycode |
|------|------------|-------------|-------------|---------|
| up | 410 | 310 | 510 | 103 |
| down | 750 | 650 | 850 | 108 |
| left | 1020 | 920 | 1120 | 105 |

| | | | | |
|-------|------|-------------|------|-------|
| right | 1200 | 1121 | 1300 | 106 |
| ok | 1440 | 1330 | 1530 | 0x160 |
| exit | 1740 | 1640 | 1840 | 174 |

备注:

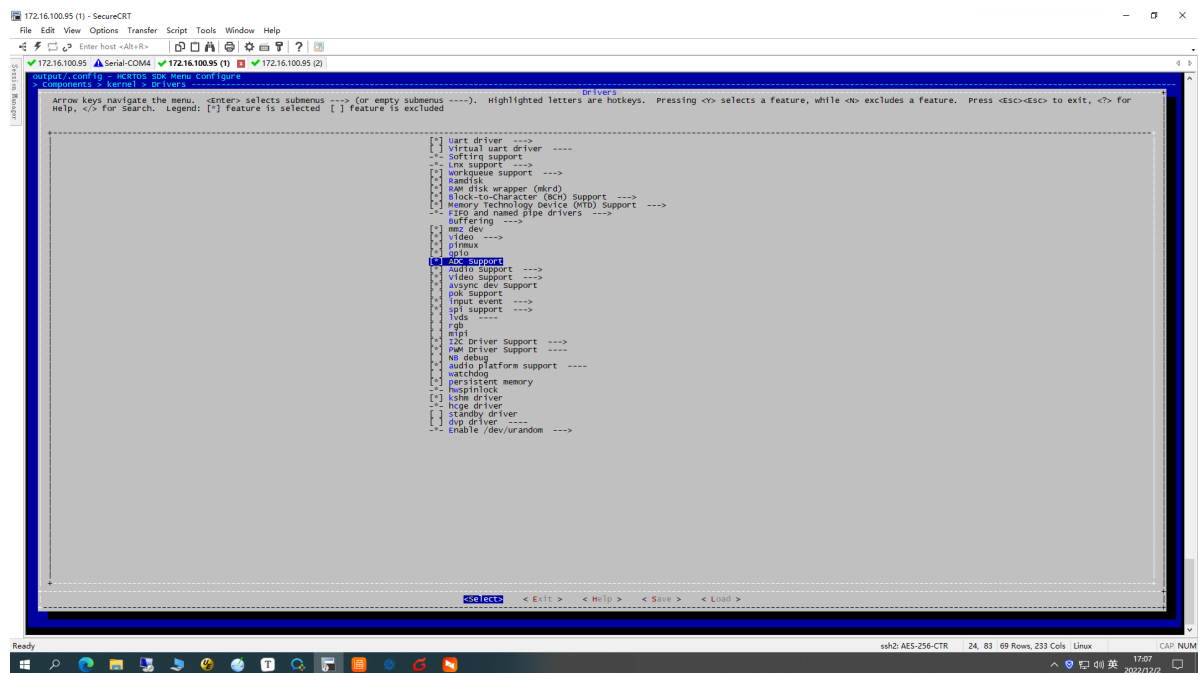
1. keycode值请参考/components/kernel/source/include/uapi/hcuapi/input-event-codes.h
2. 每个按键的误差范围为1.8v的6%，即108mv，这里是 **正负100**，请测量板子实际的电压值，以板子实际情况为准
3. 按键之间的范围不能重叠

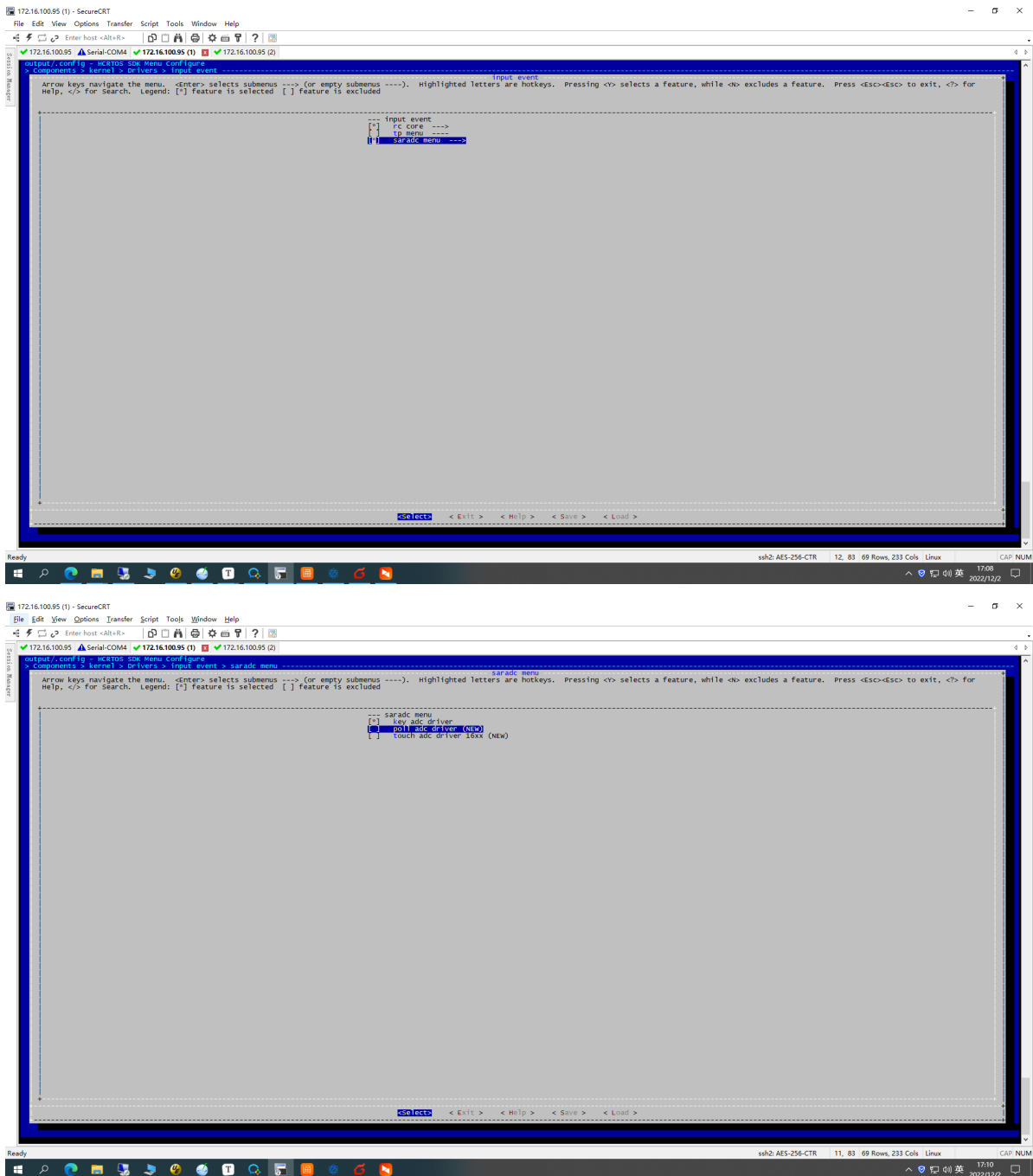
3.配置dts和kernel config

1. 配置dts，在hc16xx-db-d3100-v20.dts中增加adc key配置

```
&key_adc0 {
    status = "okay";
    /*key_map = <voltage_min, voltage_max, keycode>*/
    key-map = <310 510 103>,
              <650 850 108>,
              <920 1120 105>,
              <1121 1300 106>,
              <1330 1530 0x160>,
              <1640 1840 174>;
};
```

2.配置kernel config





至此按键修改完成，编译内核，重新烧写固件即可。

4.测试

使用getevent来判断打印的keycode与dts中设置的keycode是否一致。

12.16 如何在boot启动阶段拉高或拉低gpio

以b100为例，需要在文件 `\board\hc15xx\common\dts\hc15xx-db-b100-hcdemo.dts` 中增加以下节点：

```
gpio-out-def {
    status = "okay"; // 设置为okay即生效， disable即无效。
    gpio-group = <PINPAD_L00 GPIO_ACTIVE_HIGH PINPAD_R03 GPIO_ACTIVE_LOW>; //
    修改： 根据需要将对应的pin脚拉高或者拉低。
};
```

保存好修改后，进行重新编译：

```
make o=b1 kernel-rebuild all
make kernel-rebuild all
```

12.17 固件升级

12.17.1 hcfota介绍

hcfota是hichip固件升级程序，其升级方式有：

- usb device，待支持，使用PC工具通过usb device进行升级
- U盘升级，已支持，通过U盘进行升级
- sd 卡升级，已支持，通过sd卡进行升级
- network，待支持，通过网络进行升级

支持的方案有：

- nor flash only，已支持
- spi-nand only，待支持
- nor+emmc，待支持
- nor+nand，待支持

进入升级模式的方法可以有：

- adc_key升级键：开机时，长按升级键，即可进入升级模式。升级键在dts中定义
- 检测插入U盘，mount成功后，遍历U盘中的升级文件，然后自动升级。
- hcfota接口：app中通过菜单直接调用hcfota接口

升级固件：

HCFOTA_HCxxxxx_xxxxx.bin：系统固件

12.17.2 hcfota实现原理

hcartos系统阶段，app发起升级请求，hcfota会去读取U盘中的HCFOTA_HCxxxxx_xxxxx.bin文件，然后进行升级。

12.17.3 hcfota支持的方案

公版板型，hcdemo和projector都未配置hcfota，需要用户自己配置

12.17.4 hcfota相关代码介绍

代码

```
\components\hcfota  
//hcfota源码
```

12.17.5 hcfota配置

1.板级配置

如果需要adc按键升级，可以参照如下修改，修改hc16xx-db-d5200-v11.dts

```
key-adc@0 {  
    status = "disabled";  
    key-num = <6>;                //按键的数量  
    /*key_map = <voltage_min, voltage_max, code>*/  
    key-map = <200 500 103>,  
             <501 800 108>,  
             <901 1000 105>,  
             <1101 1200 106>,  
             <1301 1500 0x160>,  
             <1600 1750 174>;  
};  
  
hcfota-upgrade {  
    status = "disabled";        //使能  
    adc_key = <103>;            //升级键键值，需是input-event-codes.h定义的keycode  
};
```

components\kernel\source\include\uapi\hcuapi\input-event-codes.h

12.17.6 hcfota调试：app如何调用hcfota接口

hfota头文件位于：components\hcfota\source\hcfota.h

可以参考components\cmds\source\hcfota\hcfota_test.c

app上的流程，请参考setup.c里的函数 `software_update_event`。

12.8 如何打开cjc8988?

cjc8988芯片是一颗超低功耗的双路ADC和DAC的音频编解码器,有2个耳机放大器或立体声输入输出接口的AD/DA转换器

以C3000为例:

12.8.1 硬件要求:

1. 烧写完成后, **拔出ejtag**, 这点很重要!
2. I2S_IN JP4的跳线帽插上 ---- 跳线帽拔出时可以用ejtag, 插入时可以用I2S。
3. AU1或者AU2 连上speaker。

12.8.2 软件要求:

1. I2C @3的baudrate改成115200。 如果是其他配置, 需要确认该芯片是挂在哪个I2C总线上。
2. 使用c3000默认配置。

12.8.3 测试方法

1. 插入U盘。待识别到盘符。
2. 串口中运行: mp play /media/sda1/0000520223111124484.mp4, 其中 0000520223111124484.mp4为U盘中存在的音视频文件。
3. 除了方法2, 也可以在界面上选择播放文件进行播放

12.8.4 测试结果

```
hc1600a@dbc3000v10# mp play /media/sda1/0000520223111124484.mp4
hcplayer_create: /media/sda1/0000520223111124484.mp4
read_thread start
create_io_ctx
open_io_ctx
hc1600a@dbc3000v10# name mov,mp4,m4a,3gp,3g2,mj2
open_io_ctx exit
parse_stream_info
[mov,mp4,m4a,3gp,3g2,mj2 @ 0x81514800] All info of 5 streams found
ic->nb_streams 5
stream_component_open codec_id 86018
try_ffmpeg_decoder 1
[aac_fixed @ 0x815b5400] warning: not compiled with thread support, using thread
emulation
d->avctx 0x815b5400
stream_component_open codec_id 27
try_ffmpeg_decoder 0
video extradata_size 40
/dev/viddec hd1 0x81591760
is->video_stream 0, is->audio_stream 2, is->subtitle_stream -1
parse done, enter pkt read loop
hcplayer_emit_msg type 3, msg_id -2125455360
hcplayer_emit_msg type 4, msg_id -2125455360
[aac_fixed @ 0x815b5400] This stream seems to incorrectly report its last channel
as SCE[1], mapping to LFE[0]
frame->channels 6, frame->channel_layout 0x3f
audio info is changed, restart auddec
acfg->extradata_size 0
ctx->block_align 0, ctx->bitdepth 16
ctx->block_alignbuffering 0
hcplayer_emit_msg type 4, msg_id -2125455360
hcplayer_emit_msg type 1, msg_id -2125455360
av play in
av play out
params->start_threshold 12
cjc8988 i2c3 fd 13
```

//可以看到如下打印

```
cjc8988 i2c3 fplayer playing
chip_addr 0x2: val 23 is right
chip_addr 0x14: val 255 is right
chip_addr 0x34: val 248 is right
cjc8988 i2c config done!
video underrun
```

// 确认I2C配置是对的，有声音出来。

```
    Last message repeated 1 times
hcplayer_emit_msg type 4, msg_id -2125455360
buffering 0
vol fade 0, tar 0, cur 0, ct 9
hcplayer_emit_msg type 2, msg_id -2125455360
player paused
hcplayer_emit_msg type 4, msg_id -2125455360
hcplayer_emit_msg type 1, msg_id -2125455360
av play in
av play out
buffering 100
player playing
```

12.8.5 其他类似芯片

比如cs4344，是一颗被动芯片，保证电路有波形即可。

wm8960，也需要挂在I2C总线上，与cjc8988类似。

13. 常见问题Q&A

13.1 checkout到新的branch，比如从2022.07.y更新到2022.09.y，编译不过？

一般是因为旧的编译文件没有清理导致的，请运行以下命令后再重新编译：

```
cd hcrtos
git submodule update --init --recursive
rm output_b1 -rf
rm output -rf
```

最好的方法：请重新git clone 一份代码。不要在旧的版本上checkout操作！

13.2 B200 不同版本串口RX不一样，如：

DT-B200 V1 板子，Rx为PINPAD_T00 8，这也是我们sdk中的默认配置。

```
board\hc15xx\common\dts\hc15xx-db-b200.dts
```

如果拿到的是B200 V2.1，B200 V2.2 那么需要改成PINPAD_R05 2 才行。

```
uart@1 {
    pinmux-active = <PINPAD_T00 8 PINPAD_R05 2>;
    devpath = "/dev/uart1";
    status = "okay";
};
```

改完重新编译：

```
cd hcrtos
make O=bl kernel-rebuild all
make kernel-rebuild all
```

13.3 配置好屏幕后，显示图片异常

一般是由于ejtag与屏幕的pin脚share导致的。

解决方法：硬件上将ejtag做出跳线的形式，烧录时才插上重启。验证屏幕时，烧录好软件就拔出跳线帽再启机。

13.4 提示工具链未安装？

请详细阅读本文第一章的内容！注意区分工具链的名字，建议客户可以两个都安装上。

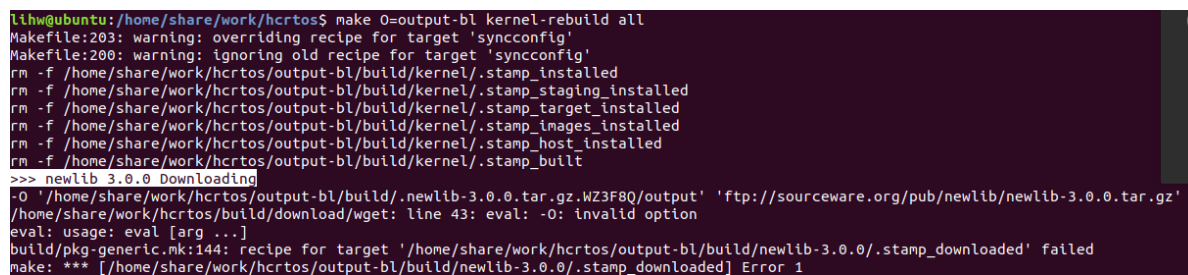
注意！！：hichip freeRTOS SDK版本 2022.09.y及之前的版本使用的是：

Codescape.GNU.Tools.Package.**2019.09-03**.for.MIPS32.MTI.Bare.Metal.Ubuntu-18.04.5.x86_64.tar.gz

2022.09.y之后的版本使用的是：

Codescape.GNU.Tools.Package.**2019.09-03-2**.for.MIPS32.MTI.Bare.Metal.Ubuntu-18.04.5.x86_64.tar.gz

13.5 编译出现wget 参数错误？



```
lihw@ubuntu:/home/share/work/hcrtos$ make O=output-bl kernel-rebuild all
Makefile:203: warning: overriding recipe for target 'synconfig'
Makefile:200: warning: ignoring old recipe for target 'synconfig'
rm -f /home/share/work/hcrtos/output-bl/build/kernel/.stamp_installed
rm -f /home/share/work/hcrtos/output-bl/build/kernel/.stamp_staging_installed
rm -f /home/share/work/hcrtos/output-bl/build/kernel/.stamp_target_installed
rm -f /home/share/work/hcrtos/output-bl/build/kernel/.stamp_images_installed
rm -f /home/share/work/hcrtos/output-bl/build/kernel/.stamp_host_installed
rm -f /home/share/work/hcrtos/output-bl/build/kernel/.stamp_built
>>> newlib 3.0.0 Downloading
-O '/home/share/work/hcrtos/output-bl/build/.newlib-3.0.0.tar.gz.WZ3F8Q/output' 'ftp://sourceware.org/pub/newlib/newlib-3.0.0.tar.gz'
/home/share/work/hcrtos/build/download/wget: line 43: eval: -O: invalid option
eval: usage: eval [arg ...]
build/pkg-generic.mk:144: recipe for target '/home/share/work/hcrtos/output-bl/build/newlib-3.0.0/.stamp_downloaded' failed
make: *** [/home/share/work/hcrtos/output-bl/build/newlib-3.0.0/.stamp_downloaded] Error 1
```

这个错误是由于编译bootloader 或者app时，使用了O=xxx，而后面重新编译时，又错误的使用了O=yyy引起的。

典型的例子就是：

刚开始编译使用了一下的命令：

```
make O=output_bl hichip_hc15xx_db_a210_v12_hcdemo_bl_defconfig
```

```
make O=output_bl all
```

后续编译又使用了如下命令：

```
make O=output-bl kernel-rebuild all
```

这里错误点在于：前面使用了下划线的 output_bl，所以生成的目录是output_bl，后面编译使用的是中划线的 output-bl。

13.6 编译提示hdmirx和usbmirror库找不到？

```
compiling src/projector/ui_rsc/image/cast/img_lv_demo_music_slider_knob_large.o
compiling src/projector/ui_rsc/image/cast/img_lv_demo_music_btn_play.o
/opt/mips32-mti-elf/2019.09-03-2/bin/mips-mti-elf-ld: cannot find -lusbmirror
/opt/mips32-mti-elf/2019.09-03-2/bin/mips-mti-elf-ld: cannot find -lvddrv_hdmirx
/home/share/work/hcrtos/components/applications/apps-projector/source/Makefile:23: recipe for target 'projector.out' failed
make[3]: *** [projector.out] Error 1
/home/share/work/hcrtos/build/Makefile.entry:35: recipe for target 'sub-make' failed
make[2]: *** [sub-make] Error 2
Makefile:16: recipe for target '_all' failed
make[1]: *** [_all] Error 2
make[1]: Leaving directory '/home/share/work/hcrtos/output/build/apps-projector'
build/pkg-generic.mk:250: recipe for target '/home/share/work/hcrtos/output/build/apps-projector/.stamp_built' failed
make: *** [/home/share/work/hcrtos/output/build/apps-projector/.stamp_built] Error 2
```

海奇SDK，默认是不release hdmirx的库和有线同屏的库的。

其中hdmirx可以通过sdk发布文档的介绍进行下载，如果提示无下载权限，请联系海奇窗口。

usbmirror 暂时未进行权限管控，如项目有用到有线同屏，请联系海奇窗口。

13.7 编译后，板子没声音出来？

如果是海奇一代芯片，需要在dts中打开以下配置：

```
pwm-dac {
    status = "okay";
};
```

打开后重新编译：

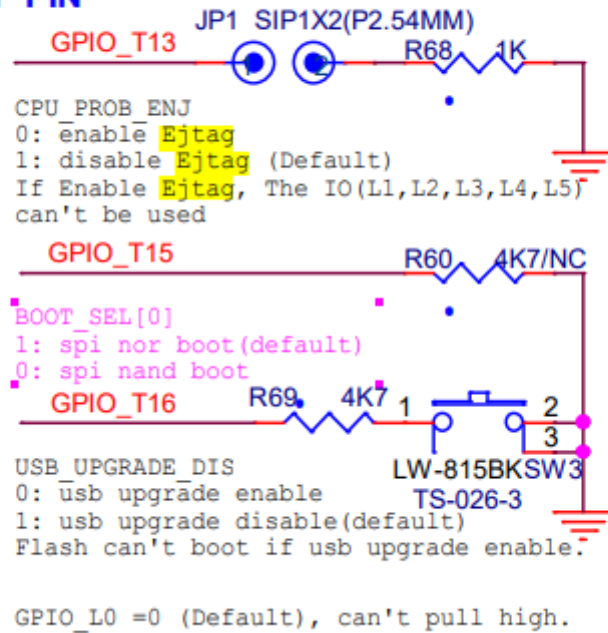
```
make O=b1 kernel-rebuild all

make kernel-rebuild all
```

如果是二代的芯片，则需要关闭ejtag。

以C3000为例，需要禁用L1、L2、L3、L4、L5这几根pin脚。

STRAP PIN



B

13.8 遇到hcrtos的SDK问题，如何报给原厂？

在开通git账户时，一般会同时给开通redmine账号，拿到账号密码后，登录以下网址，即可报问题：https://redmine.hichiptech.com/redmine/login?back_url=https%3A%2F%2Fredmine.hichiptech.com%2Fredmine%2F。

原厂工程师看到后会第一时间进行处理。

13.9 一代芯片支持nand flash & spi nand吗？

目前一代芯片硬件上不支持nand flash，SPI nand由于驱动没开发，目前也不支持，但后续可以支持。

二代芯片的linux都支持，但freertos上驱动未开发完成。