

1. 更新日志
2. 文档说明
3. 开发环境设置
4. 设置第三方软件的本地下载路径
5. 确认软件版本
6. 软件开发前准备工作
7. 软件设计和目录介绍
8. 软件配置的选择
 - 8.1 defconfig的选择
 - 8.2 dts的选择
 - 8.3 配屏文件的选择
 - 8.4 DDR配置的选择
 - 8.5 reserve
9. 编译
10. 烧录方法和工具获取
 - 10.1 烧录方法
 - 10.2 工具获取
 - 10.3 文件获取
 - 10.4 hcprogrammer烧录方式介绍
11. 烧录
 - 11.1 GDB烧录
 - 11.2 GDB在线调试
12. 快速开发指南
 - 12.1 从bootrom到hcboot
 - 12.2 压缩与解压
 - 12.3 调试串口配置
 - 12.4 驱动程序头文件
 - 12.5 无线投屏介绍
 - 12.5.1无线投屏代码介绍
 - 12.5.2 dlina接口函数
 - 12.5.3 miracast接口函数
 - 12.5.4 aircast接口函数
 - 12.5.5 wifi manager接口函数
 - 12.6 如何开启&关闭&更换boot show logo
 - 12.6.1 如何开启boot show logo
 - 12.6.1.1 如果是未编译过的工程,
 - 12.6.1.2 如果是已经编译过的工程
 - 12.6.2 如何关闭boot show logo
 - 12.6.3 如何更换boot show logo
 - 12.6.4 如何更换.h264的logo
 - 12.7 屏幕如何做到旋转。
 - 12.7.1 OSD的旋转
 - 12.7.2 视频的旋转:
 - 12.7.3 一键旋转功能
 - 12.7.4 同屏投屏时如何旋转和设置分辨率
 - 12.8 SD卡驱动
 - 12.9 LVGL最高支持的帧数配置
 - 12.10 如何在demo配置之外, 增加定制化的板级配置
 - 12.10.1 增加一个板级配置
 - 12.11 hcRTOS Nor Flash区间的分布
 - 12.11.1 流程介绍
 - 12.11.2 如何增加一个客户可读可写的分区?
 - 12.11.3 如何临时添加一个不支持的flash
 - 12.11.4 如何修改分区大小?
 - 12.11.5 如何读写flash里的OTP区域

- 12.12 uart蓝牙的配置
 - 12.12.1 串口蓝牙
- 12.13 wifi的支持
- 12.14 TP驱动的集成和测试
- 12.15 hdmi in调试
- 12.16 OSD 图层介绍和分辨率的修改
- 12.17 VIDEO图层的介绍和使用、测试
- 12.18 按键类支持
 - 12.18.1 如何增加ir遥控器
- 12.19 如何在boot启动阶段拉高或拉低gpio
- 12.20 固件升级
 - 12.20.1 hcfota介绍
 - 12.20.2 hcfota实现原理
 - 12.20.3 hcfota支持的方案
 - 12.20.4 hcfota相关代码介绍
 - 12.20.5 hcfota配置
 - 12.20.6 hcfota调试: app如何调用hcfota接口
- 12.21 如何打开cjc8988?
 - 12.21.1 硬件要求:
 - 12.21.2 软件要求:
 - 12.21.3 测试方法
 - 12.21.4 测试结果
 - 12.21.5 其他类似芯片
- 12.22 梯形校正
- 12.23 boot standby 配置说明
- 12.24 如何添加adc key支持
- 12.25 pin脚的功能查询
- 12.26 串口读写sample
- 12.27 配置GPIO, 使其在bootloader阶段就生效。
- 12.28 配置I2C sample
- 12.29 如何修改edid进行测试
- 12.30 PQ工具使用说明
- 12.31 如何使用RTC?
- 12.32 USB gadget相关驱动说明
- 12.33 watch dog相关说明
- 12.34 PHY相关说明
- 12.35 如何在SDK中集成第三方库?
- 12.36 gsensor的集成和使用
- 12.37 hcrtos死机时调试小方法
- 12.38 投影仪声音调节的方法
- 12.39 配屏
- 12.40 图片解码的规格
- 12.41 airp2p配置常见问题
- 12.42 如何离线更换image里的logo文件?

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 发现苹果同屏不可用
- 13.8 编译后, 板子没声音出来?
- 13.9 遇到hcrtos的SDK问题, 如何报给原厂?
- 13.10 一代芯片支持nand flash & spi nand吗?
- 13.11 如何增加 简易的 测试api 或者 复现问题 的代码?
- 13.12 如何打patch?
- 13.13 客户代码如何打包成库文件

- 13.14 如何提升开机时间?
 - 13.14.1 影响开机时间的因素主要有以下几点:
 - 13.14.2 主要调试手段:
 - 13.14.3 主要方法
- 13.15 reserve

1. 更新日志

版本号	日期	作者	内容
V1.0	2022.09.01	Finn.Wei	初始发布
V2.0	2023.09.18	Finn.Wei	重构内容
v2.1	2023.09.20	Finn.Wei	增加打patch方法 移动客制化配置保存说明到<编译>章节
v2.2	2023.10.16	Finn.Wei	修改错误并增加内存测试等使用方法。
v2.3	2023.11.16	Finn.Wei	1、增加如何临时修改edid进行测试的方法。 2、增加客户代码打包成库的方法 等。 3、增加编译时安装的工具：sudo apt-get install automake libtool 4、14.14增加如何配置快速开机的建议
V2.4	2024.01.04	Finn.Wei	增加重编译命令的详细说明 增加output/image下各个文件夹的文件介绍。详见第10章.
V2.5	2024.05.11	Finn.Wei	增加文档索引。
V2.6	2024.07.05	Finn.Wei	增加p2p配置方法。 调整hcprogrammer烧录方式索引文档。 增加离线更换logo的方法说明。

2. 文档说明

该文档适用于在海奇HC-RTOS平台上进行项目开发的软件工程师，其具备一定的软件开发能力，对硬件平台有一定的了解。

3. 开发环境设置

- 参考SDK发布邮件中的：《Hichip软件编译文件下载信息.zip》进行环境搭建。
- 看到该文档，默认客户下载环境已经搭建。如有未收到发布邮件的情况，请按以下方式搭建或联系窗口：
- 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
$ sudo apt-get install automake libtool

```

海奇的hcrtos 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。

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

4. 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，则可以用其他的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。

5. 确认软件版本

1. 如何确认已经切换到指定的SDK版本了？

在下载SDK服务器中，输入

```
git branch -av
```

例如 `hertos-2023.05.y`，以下注释部分一致，则说明本地下载的版本与远程服务器的是一致的。不同的SDK版本以此类推。

```
finn.wei@hichip01:hertos_orig$ git branch -av
* herτος-2023.05.y                                96c092e tag-hertos-2023.05.y.3 (Update
submodule commits)                                // 看 96c092e 这里，找到以下一致的，即可看到目前下载的是哪个
版本。
main                                                9d604e1 Initial commit
remotes/origin/HEAD                                -> origin/main
... //此处省略一些版本，下同。
...
remotes/origin/hertos-2023.02.y                    81b73d6 tag-hertos-2023.02.y.2
```

```
remotes/origin/hcrtos-2023.04.y      3435d81 tag-hcrtos-2023.04.y.5
...
remotes/origin/hcrtos-2023.05.y      96c092e tag-hcrtos-2023.05.y.3 (Update
submodule commits) //上面的96c092e与此处（远程服务器）的 96c092e是一样的。说明目
前的SDK版本（即SDK分支）为hcrtos-2023.05.y，且目前处在tag: tag-hcrtos-2023.05.y.3上。
...
remotes/origin/main                  9d604e1 Initial commit
```

6. 软件开发前准备工作

1. 拿到板子，确认板子是否有发PCB和原理图给海奇硬件terry.lin@hichiptech.com确认过。保证板上电正常。
2. 确认板子芯片型号。
3. 确认板子DDR型号: DDR2还是DDR3? 容量大小?
4. 确认板子输出是HDMI还是屏幕? 屏幕型号: RGB/LVDS/MIPI?
5. 确认板子外围功能，如：有无wifi? 有无sd卡? 有无蓝牙? 有无网口等等
6. 确认板子电源电压，串口rx、tx，确保板子连接正确。

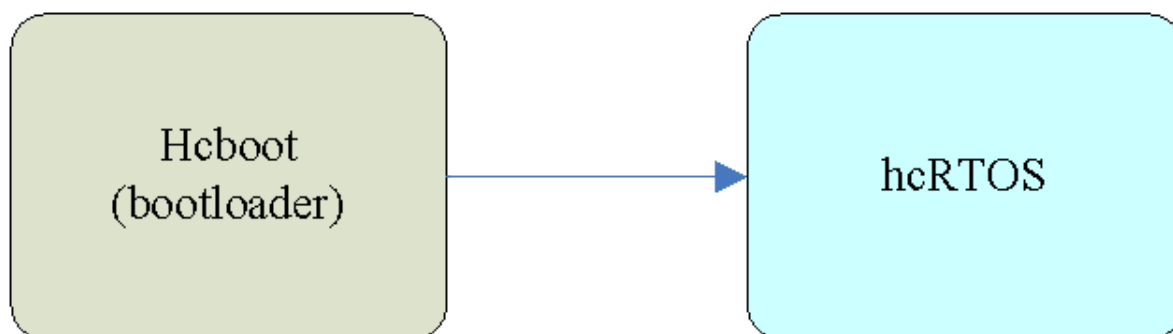
7. 软件设计和目录介绍

1. 软件相关设计

海奇hcRTOS系统应用是采用Hcboot来引导启动的。

Hcboot基于hcRTOS开发平台进行开发，但又独立于hcRTOS，经过深度裁减后用于引导和启动hcRTOS系统固件。

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

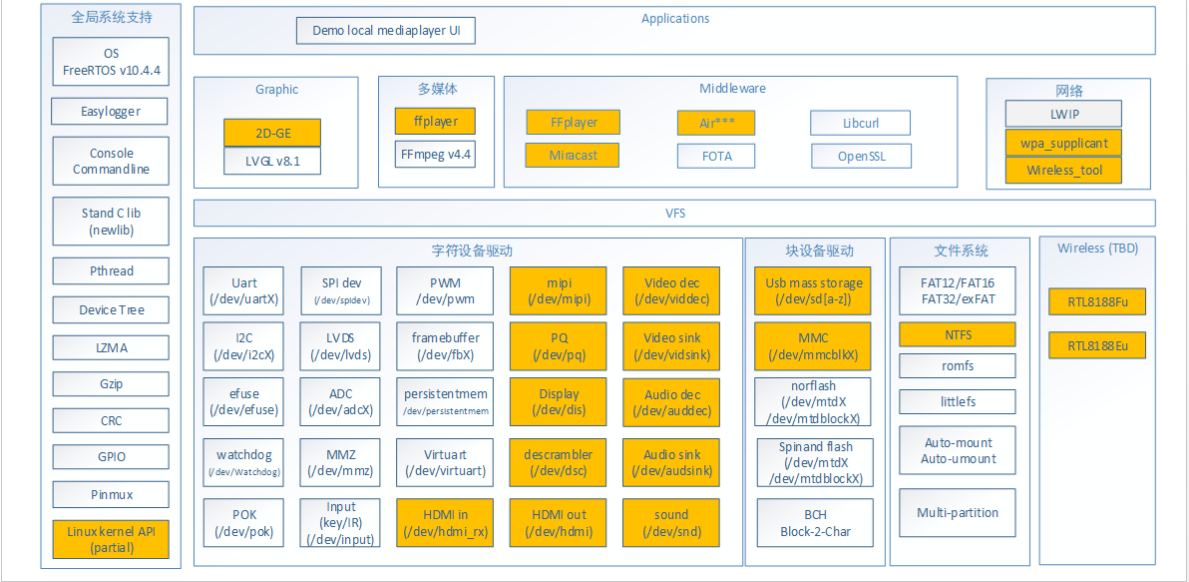


Hcboot经过深度裁剪后，剩余引导hcRTOS的功能、show logo功能、升级检测功能等一些基本的功能，也可以认为是一个小应用，开发者也可以进行响应的开发。

hcRTOS是完整功能的SDK，开发者可以按项目需求进行深度开发。其详细架构图如下所示：

hcRTOS Software Stack

Open API
Open Source



2. SDK目录简单介绍

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

```
.
├── board
//板级配置
|   ├── hc15xx
//海奇一代芯片板级配置
|   ├── hc16xx
//海奇二代芯片板级配置
|   └── 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                                        //
lvg1 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_bl_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_bl_defconfig
//d3101的bootloader配置
    ├── hichip_hc16xx_cb_d3101_v20_projector_c1_cj01_defconfig
//d3101的app配置。

```

```

├─ dl
|   ├── 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
|   ├── HcProgram_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_bl
//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

```

8. 软件配置的选择

8.1 defconfig的选择

1. 软件顶层总配置在 hcrtos/configs 目录，可以使用 `ls configs` 进行查看。然后根据第6章的准备工作，找到相应的芯片型号和项目需求，选择对应的defconfig

配置文件	说明
hcrtos/configs/hichip_hc16xx_db_d3100_v30_projector_cast_bl_defconfig	用于生成hcboot的配置文件
hcrtos/configs/hichip_hc16xx_db_d3100_v30_projector_cast_defconfig	用于生成固件

2. 配置文件命名规则：

ddr文件的选择主要有以下四个方面需要注意：

1. ddr2还是ddr3?
2. 容量大小是多少?
3. 频率用多少?
4. gdb用的ddr文件如何选择?

ddr的文件路径位于： `\board\hc16xx\common\ddrinit`。

gdb工具选用的ddr文件路径位于： `\board\hc16xx\common\ddrinit\gdb`。

开发者需要特别注意这两者的区别，gdb路径中的ddr文件仅用于gdb工具烧录和调试用。

ddr文件在软件中的配置位于8.1节介绍的defconfig中，例如；

`hichip_hc16xx_db_d3100_v30_projector_cast_defconfig` 中有如下定义：

```
BR2_EXTERNAL_BOARD_DDRINIT_FILE="${TOPDIR}/board/hc16xx/common/ddrinit/hc16xx_ddr3_128M_1066MHz.abs"
```

```
hc16xx_ddr3_128M_1066MHz.abs
|      |      |      |----- 文件后缀。
|      |      |-----  ddr频率。
|      |-----  ddr容量大小。
|-----  ddr2 或者 ddr3。
|-----  海奇一代芯片为hc15xx，二代芯片为
hc16xx。
```

!!! 重点：开发者进行开发的过程中，需要注意拿打样的板子进行overnight烧机测试，以验证DDR的稳定性。也可使用如下方法来测试ddr：

按如下两图进行 `make menuconfig` 的配置，

```
Input/.config - HCR10S SDK Menu Configure
Components
Components
BR2_PACKAGE_MEMTESTER:

A utility for testing the memory subsystem for
faults.

http://pyropus.ca/software/memtester/

Symbol: BR2_PACKAGE_MEMTESTER [=n]
Type   : bool
Prompt: memtester
Location:
    -> Components
Defined at memtester:1
```

nus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a
ed [] feature is excluded

```
( ) Prebuilt subdir
[*] prebuilts --->
    Applications Configuration --->
[ ] Stack Smashing Protector(ssp)
-*- Newlib --->
[*] kernel --->
[*] Cms --->
-*- pthread
-*- ffmpeg --->
[*] hc-examples
[ ] hdmi wireless example
[*] Opencore-amr library
[*] libusb
[ ] build libusb examples
[*] hcfota
-*- liblvgl --->
[ ] bluetooth ----
[ ] quicklz
[ ] liblzo
-*- Zlib
[*] libopenssl --->
[ ] mbedtls
[*] cJSON
[ ] hccast ----
[*] libcurl
[ ] curl binary
[ ] verbose strings
[*] proxy support
[*] cookies support
[*] enable extra protocols and features
    SSL/TLS library to use (None) --->
[ ] Unity Test
[ ] memtester
[*] freetype
[ ] pcre
[ ] speex
↓(+)
```

配置完后重新编译：

```
make all
```

测试方法：重编并烧写固件后，在串口执行memtester 20M。测试命令会对内存进行反复读写，这样烧机24小时，如果没有报错，则说明ddr是比较稳定的。

8.5 reserve

9. 编译

开发者进行编译前，请一定仔细阅读前面8章的内容。否则请回头仔细阅读。

1. 编译方法

如上介绍，编写者手上的板子是D3100 V30的，且项目是投影仪，带无线同屏功能，那么编译的命令是：

```
make O=output-bl hichip_hc16xx_db_d3100_v30_projector_cast_bl_defconfig
make O=output-bl all
make hichip_hc16xx_db_d3100_v30_projector_cast_defconfig
make all
```

!!! 这里要注意的是：编译bootloader可以指定目录，但编译主应用不可以指定编译路径。

2. 常用的编译命令

修改了bootloader和dts，需要按如下命令进行重新编译：

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

进行bootloader的配置：

```
make O=output-bl menuconfig
```

进行app的配置选项：

```
make menuconfig
```

修改了音视频的支持格式，特别是删除某些音视频格式支持，需要用如下命令：

```
make kernel-reconfigure kernel-clean kernel-rebuild all
```

修改了应用，需要如下命令：

```
make apps-projector-rebuild all
```

修改了components/hccast下的内容，需要：

```
make hccast-reconfigure hccast-rebuild all
```

删除全部编译内容：

```
rm output* -rf
```

修改了components/cmds下的内容：

```
make cmds-rebuild all
```

修改了lvgl的配置：

```
make liblvgl-menuconfig
```

删减了lvgl的某些配置：

```
make liblvgl-reconfigure
```

修改了components/liblvgl的代码：

```
make liblvgl-rebuild all
```

...

3. 编译完成后，会在 /output/images 下生成如下个文件。

```
$ tree output/images
```

```
output/images/
```

```
├─ for-debug //用于RD debug用的，这里面的hcfota.bin包含了全部的文件，而且会忽略版本检查，强制升级。ini文件用于ejtag升级
```

```
│   └─ hc16xx_jtag_updater.out
```

```
│   └─ HCFOTA.bin
```

```
│   └─ HCFOTA.bin.652fc
```

```
│   └─ HCFOTA_HC16C3000V10_2401021147.bin
```

```
│   └─ HCPprogrammer.exe
```

```
│   └─ sfburn.ini
```

```
├─ for-factory //spinorflash.bin是用于生成烧录器用的，即贴片生产用的。是flash上对应的烧录文件，用烧录器进行烧录。
```

```
│   └─ spinorflash.bin
```

```
├─ for-net-upgrade //这是用于网络升级的， 这里面的hcfota.bin与 "for-upgrade"里面的是一样的。 只是这个文件夹还生成了网络配置文件HCFOTA.jsonp
```

```
│   └─ HCFOTA.bin
```

```
│   └─ HCFOTA.jsonp
```

```
├─ for-upgrade //hcfota.bin“不会忽略版本检查”，“不会包含boot分区”。如需升级boot分区，需要手动执行HCFota_Generator.exe，生成对应的HCFOTA.bin
```

```
│   └─ hc16xx_jtag_updater.out
```

```
│   └─ HCFOTA.bin
```

```
|   ├── HCFOTA.bin.d02d1
|   ├── HCFOTA_HC16C3000V10_2401021147.bin
|   ├── HCPprogrammer.exe
|   └── sfburn.ini
```

4. 如何保存客户独有的配置？

一般客户拿到板子后，需要配板，会用 `make O=output-bl menuconfig` 或者 `make menuconfig` 进行配置，退出保存后，起始是保存在 `output-bl/.config` 或者 `output/.config` 下，这时，如果运行 `rm output* -rf` 删除了中间编译文件，那么配置文件也就删除了，这样就需要重新配置一次。不利于项目开发

所以，客户可以在配置完成，并验证该配置可以工作后，使用以下命令来保存配置：

```
cp output-bl/.config config/xxx_bl_defconfig
cp output/.config config/xxx_defconfig
```

配置文件命名方式可以参照第8章第1节海奇现有的统一命名方式，也可使用自定义的符合linux标准的命名方式。

这样保存后，后续运行 `rm output-bl/ output/ -rf` 命令后重编译，你只需用以下的命令就可以编译成你已经配置好的软件：

```
make O=output-bl xxx_bl_defconfig
make O=output-bl all
make xxx_defconfig
make all
```

10. 烧录方法和工具获取

10.1 烧录方法

海奇hertos支持多种烧录方法，开发者可以选择合适自己的方法。

1. gdb烧录：

优点：开发调试方便，便于跟踪代码，查看疑难问题。开发阶段必备。

缺点：需要占用比较多的pin脚，需要硬件设计时预留。

2. usb烧录

优点：较gdb更轻量化。有USB口即可实现。

缺点：仅二代芯片可用。

3. hcfota升级

优点：U盘升级，随插随升。

缺点：仅适用于板子内有程序，且支持hcfota升级，能正常启机的情况，空片无法升级。

4. 串口烧录

优点：在没有usb口的板子上容易实现。

缺点：速度慢，目前不够稳定，不推荐开发阶段使用。

5. hcprogrammer烧录

优点：只用一条usb公对公的线即可完成，目前主推的烧录方式。

缺点：hc15xx系列芯片不能烧录空片。

6. 贴片法

优点：通用烧录器进行烧录，适合于大批量生产，使用的文件路径位于：\output\images\for-factory\spinorflash.bin

缺点：需要拆下flash后才能升级。

10.2 工具获取

gdb工具购买链接：<https://item.taobao.com/item.htm?spm=a1z10.3-c.w4023-24683637015.4.45ab1f8aAXjyCm&id=696858305604>

10.3 文件获取

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）

参考视频：hclinux_gdb烧写固件教学视频.mp4

10.4 hcprogrammer烧录方式介绍

文件获取：网站：[固件烧写 · main · sw / Tools · GitLab \(hichiptech.com\)](#)

更建议用以下方式获取：

```
git clone https://gitlab.hichiptech.com:62443/sw/tools.git
```

下载后，详细位置：\tools\固件烧写\HCProgrammer说明书20231016.pdf

11. 烧录

11.1 GDB烧录

用GDB进行调试和烧录的时候，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
```

用GDB工具下载 `hcrtos/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!
```

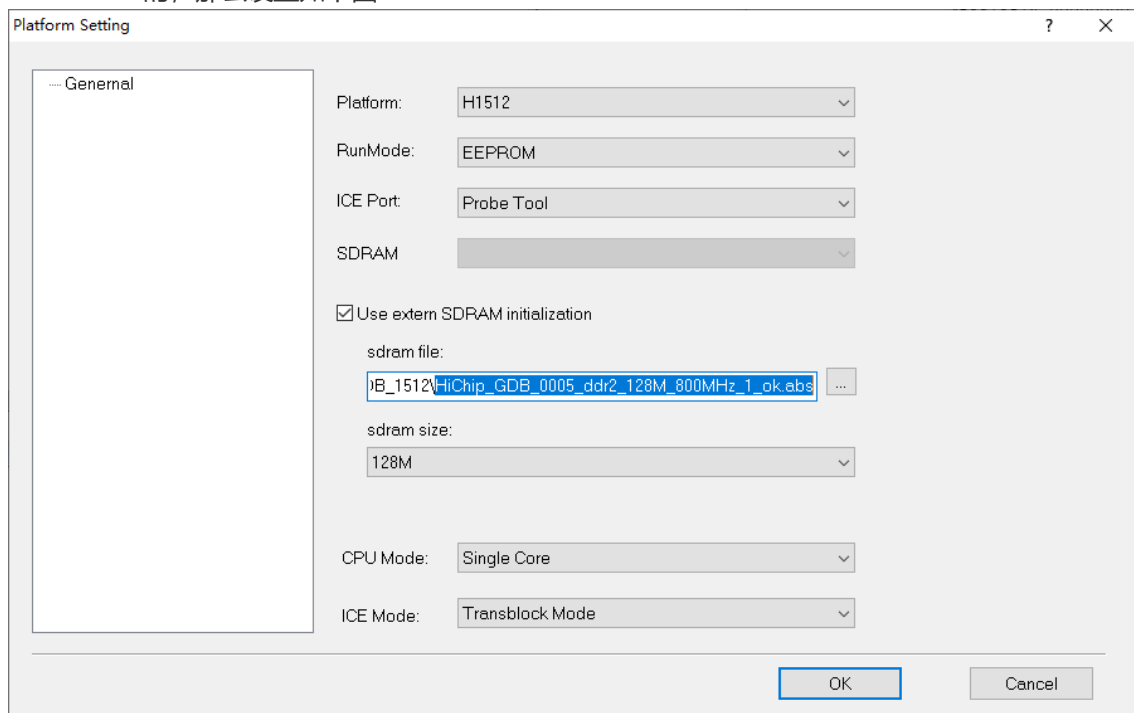
```
00190000 OK!
001a0000 OK!
001b0000 OK!
001c0000 OK!
001d0000 OK!
001e0000 OK!
001f0000 OK!
00200000 OK!
00210000 OK!
00220000 OK!
00230000 OK!
00240000 OK!
00250000 OK!
00260000 OK!
00270000 OK!
NOR FLASH Burnning Done!
```

更详细的烧录步骤，请参考第10.3章介绍的路径中的文档和视频

11.2 GDB在线调试

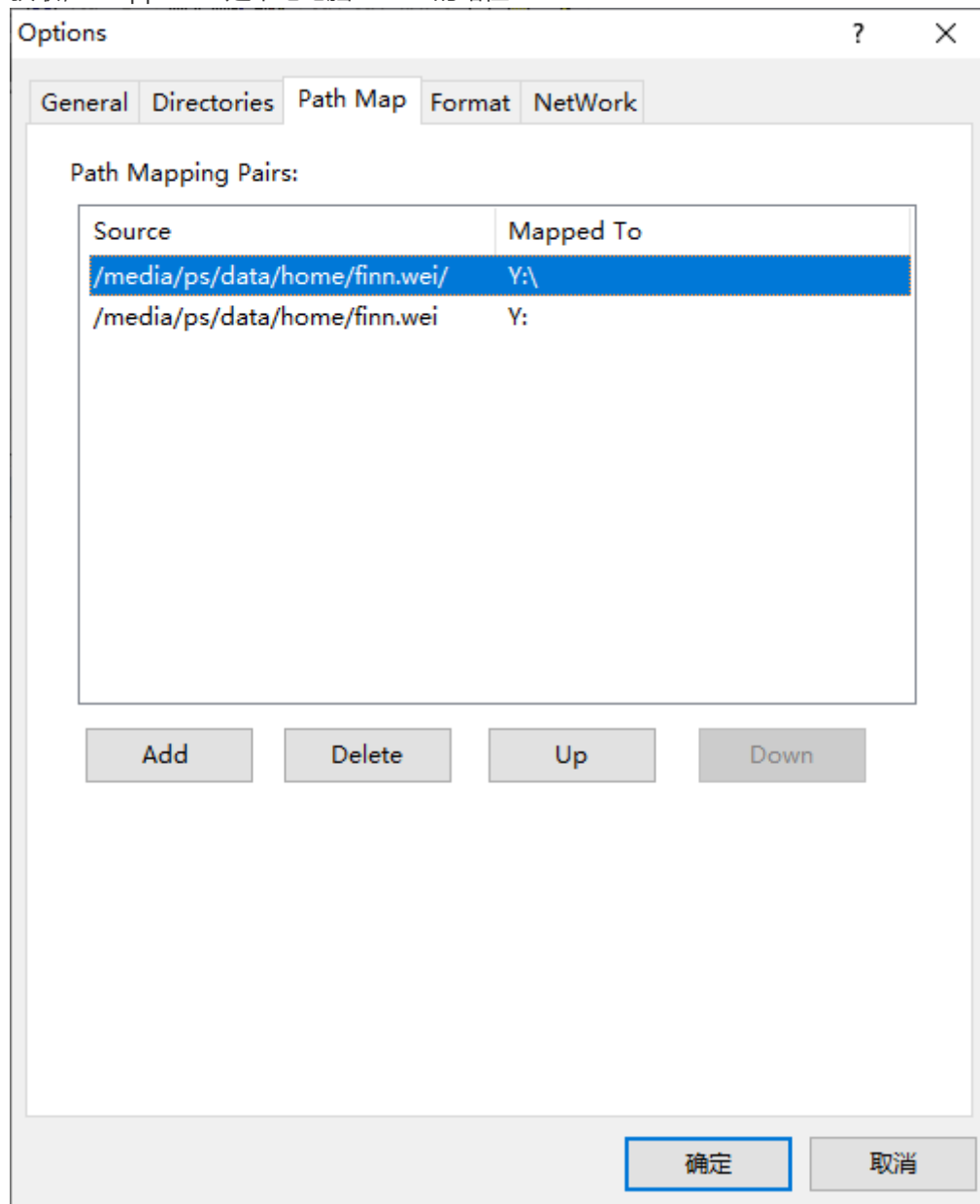
!!! 注意点:

1. 该方法是让程序实时跑在ddr上，并非从flash中load进来跑，但是也存在代码回去读flash的可能。所以需要保证flash中代码和download跑的是同一份代码编译。最好的方法就是先烧录一变再download .out跑。
2. ICE -> Platform Setting里的ddr一定要选择对，包括ddr size， ddr初始化文件，一定要对应！因为程序download跑的时候是读gdb中的ddr文件进行初始化ddr的。以我手上的A210C为例： 其是ddr2/128M的，那么设置如下图：

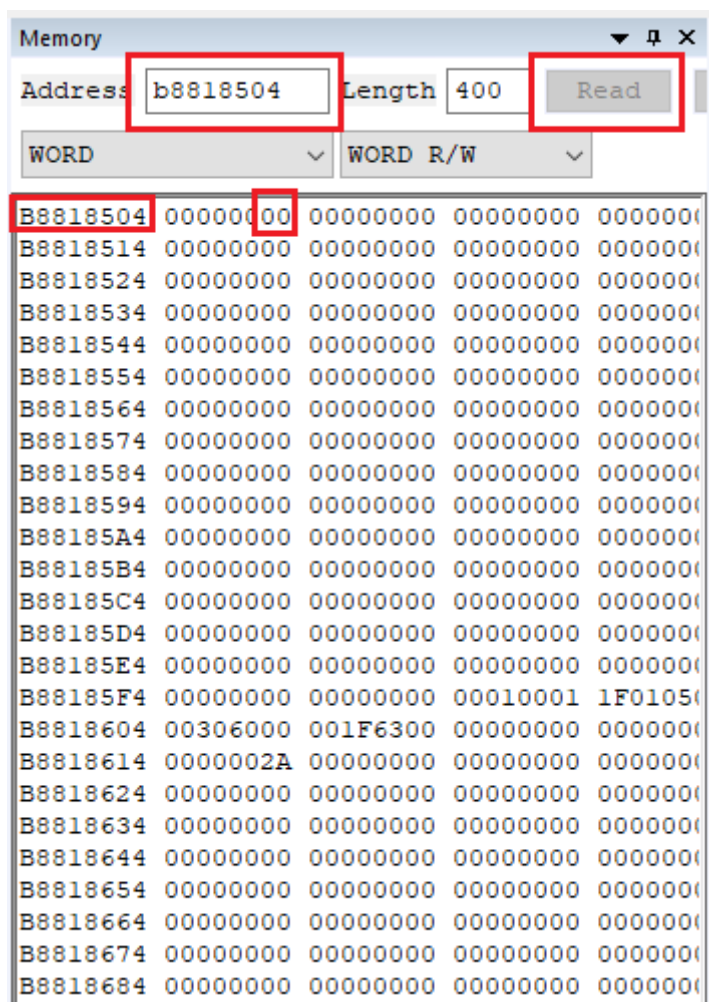


3. Tools中的Option中的Path Map一定要设置好，否则看不到源码。另外，不是自己编译的库也看不到源码。除非你要到源码后本机编译。下图中，Source是远程服务器的路径，可以通过pwd命令

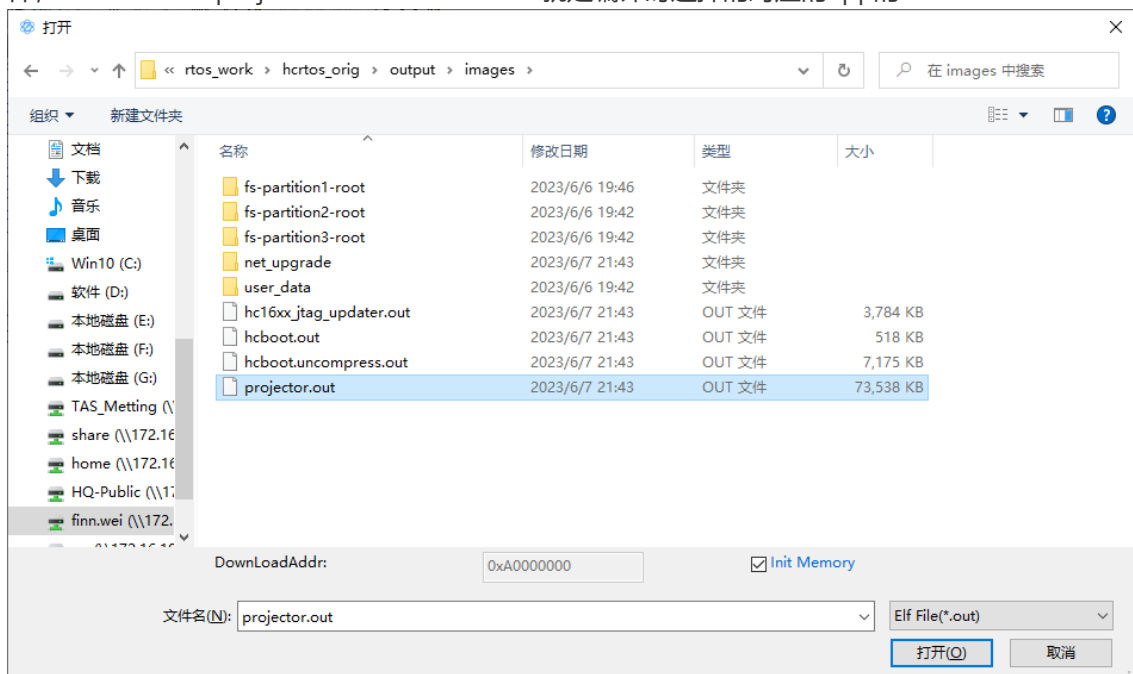
获取，Mapped To是本地电脑Samba的路径：



4. 看门狗要关闭，即在gdb的memory窗口，输入B8818504 -> 点击Read -> 将4字节的尾部改为00 --> 改完再点击下Read， 确保已经是00了。

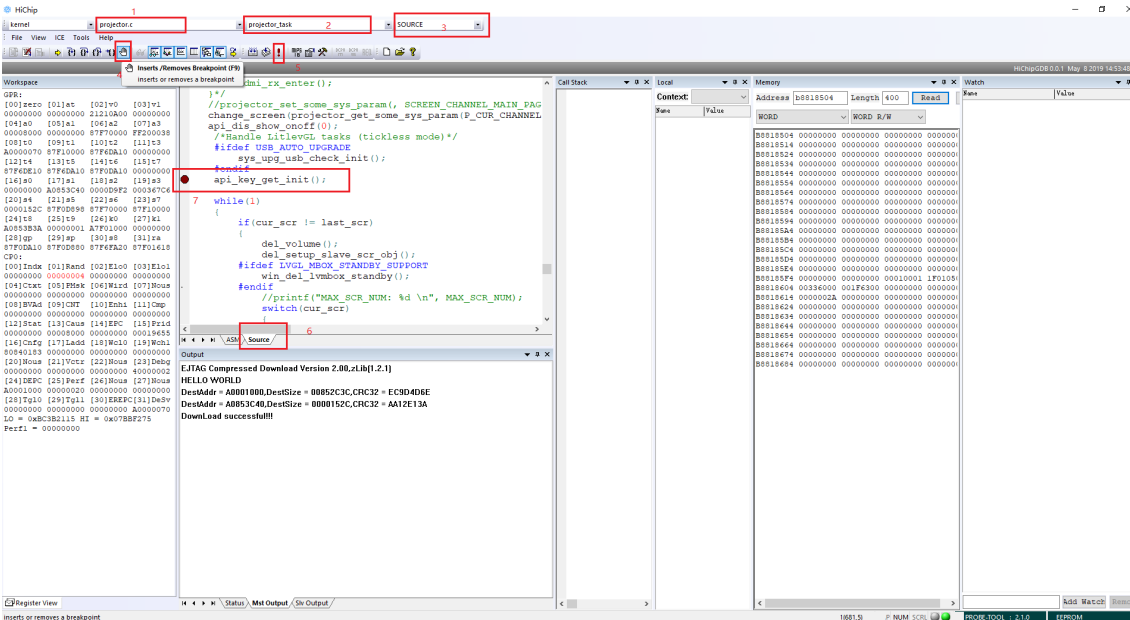


5. 用于download的.out文件位于\hcartos\output\images下，其中hcboot.out是bootload对应的文件，hcdemo.out/projector.out/hcscreen.out就是编译时选择的对应的app的.out：



6. download完.out文件，如下图所示：其中，1是文件，2是行数，3是选择源码显示，4是下断点，5是运行程序，6是源码显示框，7是下好的断点。这样下好断点，并运行后，程序就可以在断点处停下来，并在Call Stack窗口显示程序调用栈，再点击运行按钮或按F5，程序可以继续运行。也可

以选择单步运行，工具栏对应的按键，其余功能待客户自行挖掘，此处不详细介绍。



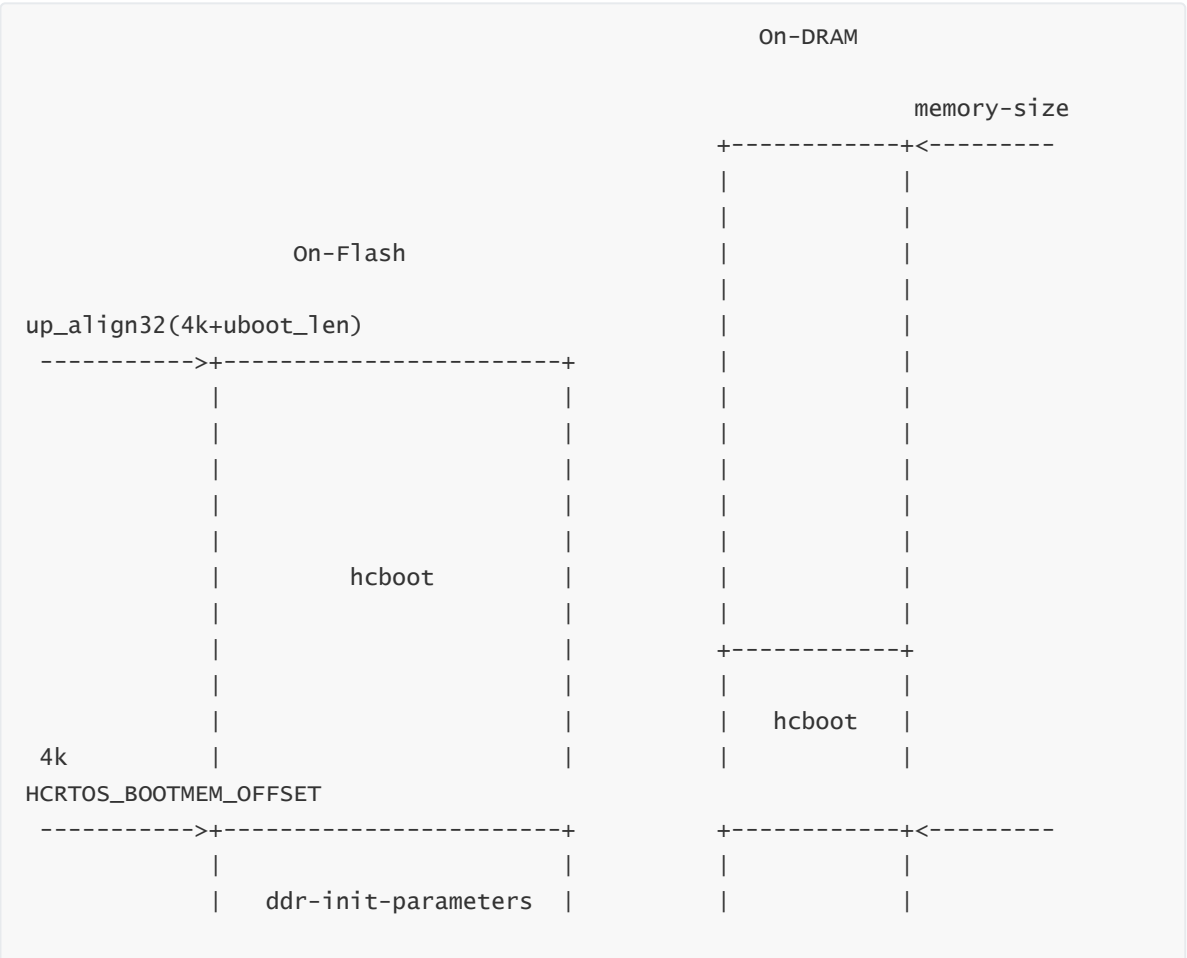
7. 参考文档《hichipgdb_userguide.pdf》

12. 快速开发指南

12.1 从bootrom到hcbboot

hcrtos/output/images/bootloader.bin 文件由hcbboot.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hcbboot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hcartos 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 hcartos/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
```

12.2 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 hcartos/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, hcartos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

12.3 调试串口配置

以 `hc16xx-db-a3100` 为例, 在 `hcartos/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), **此处需要特别注意, 硬件文档中, 串口 标号是1/2/3/4, 与代码中不同, 需要一一对应上来。**

```
$ 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 = "/hcartos/uart@3";
};
```

12.4 驱动程序头文件

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

驱动程序的头文件提供所有目前海奇开放的api给上层应用程序调用，以实现项目功能。

```
.
├─ 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

```

12.5 无线投屏介绍

12.5.1 无线投屏代码介绍

```

components/applications/apps-hcscree/  //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
    ├─ dlina                  //dlina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

```

12.5.2 dlina接口函数

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

```

hccast_dlina_event_callback    //回调函数
int hccast_dlina_service_init(hccast_dlina_event_callback func); //初始化
int hccast_dlina_service_uninit(); //注销
int hccast_dlina_service_start(); //开启dlina服务
int hccast_dlina_service_stop(); //关闭dlina服务

```

12.5.3 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
```

12.5.4 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();
```

12.5.5 wifi manager接口函数

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

wifi调试参考文档《hcartos_wifi_user_guide.pdf》。

wifi测试参考文档: gitlab: \tools\wifi测试\wifi_iperf测速使用说明文档.pdf

12.6 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
```

12.6.1 如何开启boot show logo

以 `hc1600a@dbd3100v30` 为例:

12.6.1.1 如果是未编译过的工程,

1. 需要在 `configs/hichip_hc16xx_db_d3100_v30_projector_cast_defconfig` 中配置如下:

```
BR2_EXTERNAL_BOOTMEDIA_FILE="${TOPDIR}/board/hc16xx/common/logo.m2v"
```

`configs/hichip_hc16xx_db_d3100_v30_projector_cast_b1_defconfig` 中不需要配置

`BR2_EXTERNAL_BOOTMEDIA_FILE`。

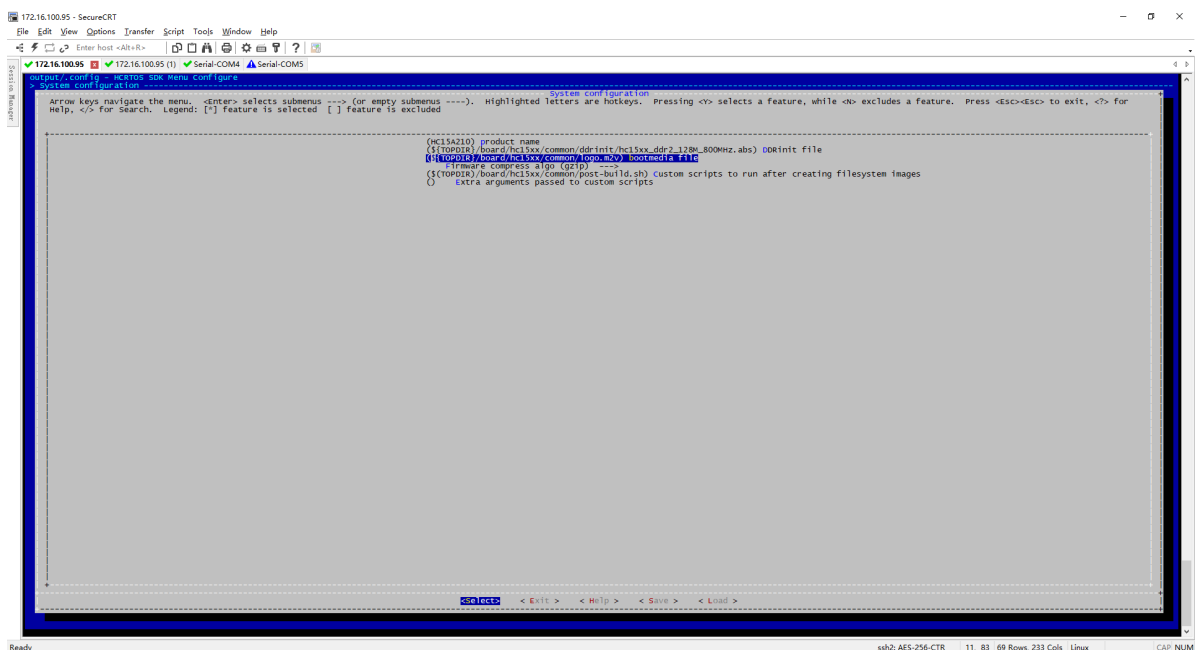
2. 修改文件: `configs/hichip_hc16xx_db_d3100_v30_projector_cast_b1_defconfig`, 打开以下句子: `CONFIG_BOOT_SHOWLOGO=y`

这样配置后, 需要如下编译

```
make O=output-b1 hichip_hc16xx_db_d3100_v30_projector_cast_b1_defconfig
make O=output-b1 all
make hichip_hc16xx_db_d3100_v30_projector_cast_defconfig
make all
```

12.6.1.2 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:

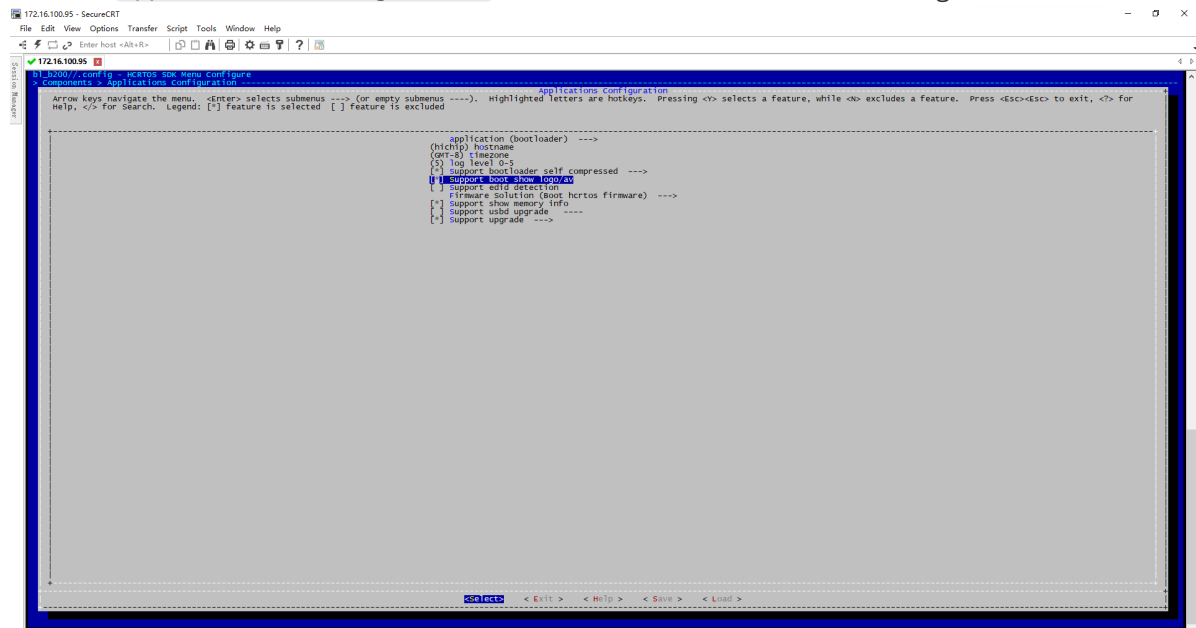


退出界面后保存:

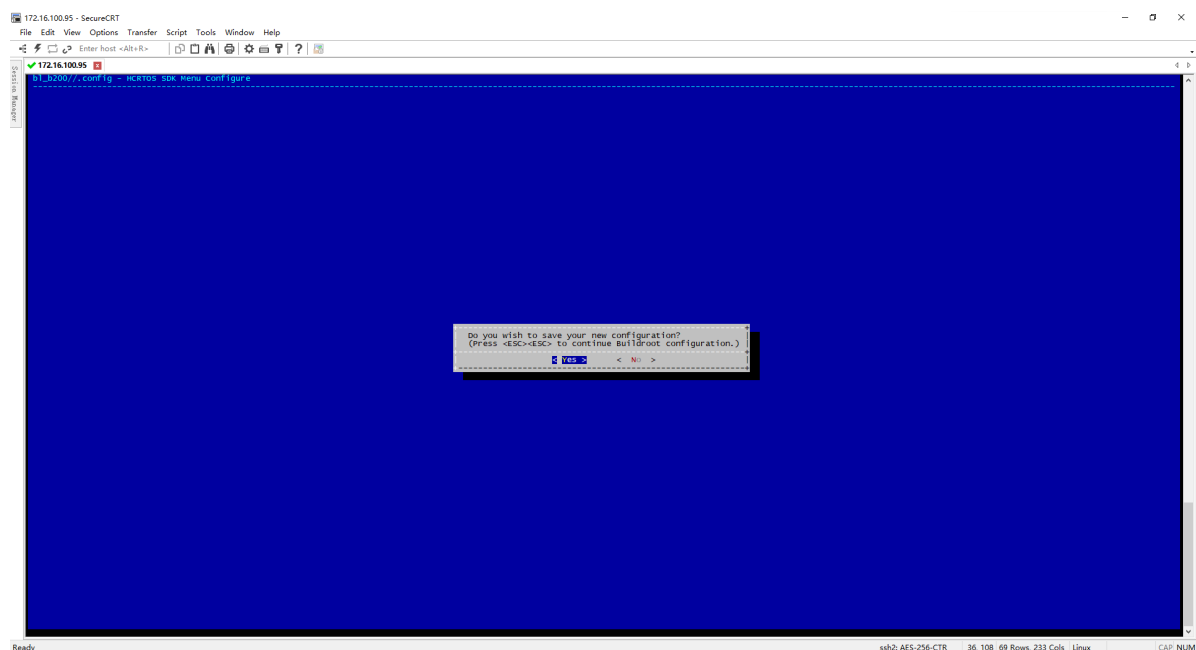
2. 进入 boot loader 的 menuconfig 进行配置:

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

12.6.2 如何关闭boot show logo

1. 参考12.7.1中开启boot show logo的配置, 进入 boot loader 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:

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

12.6.3 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>

12.6.4 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.7_z
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。

12.7 屏幕如何做到旋转。

12.7.1 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*/
};
```

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

12.7.3 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。

12.7.4 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);
```

12.8 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 {  
    //hcartos-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;  
    status = "okay";  
};
```

12.9 LVGL最高支持的帧数配置

12.10 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_d3100_v30_projector_cast_bl_defconfig  
hichip_hc16xx_db_d3100_v30_projector_cast_defconfig  
ex:  
hichip_<chip_number>_<board_name>_<feature>_bl_defconfig  
hichip_<chip_number>_<board_name>_<feature>_defconfig
```

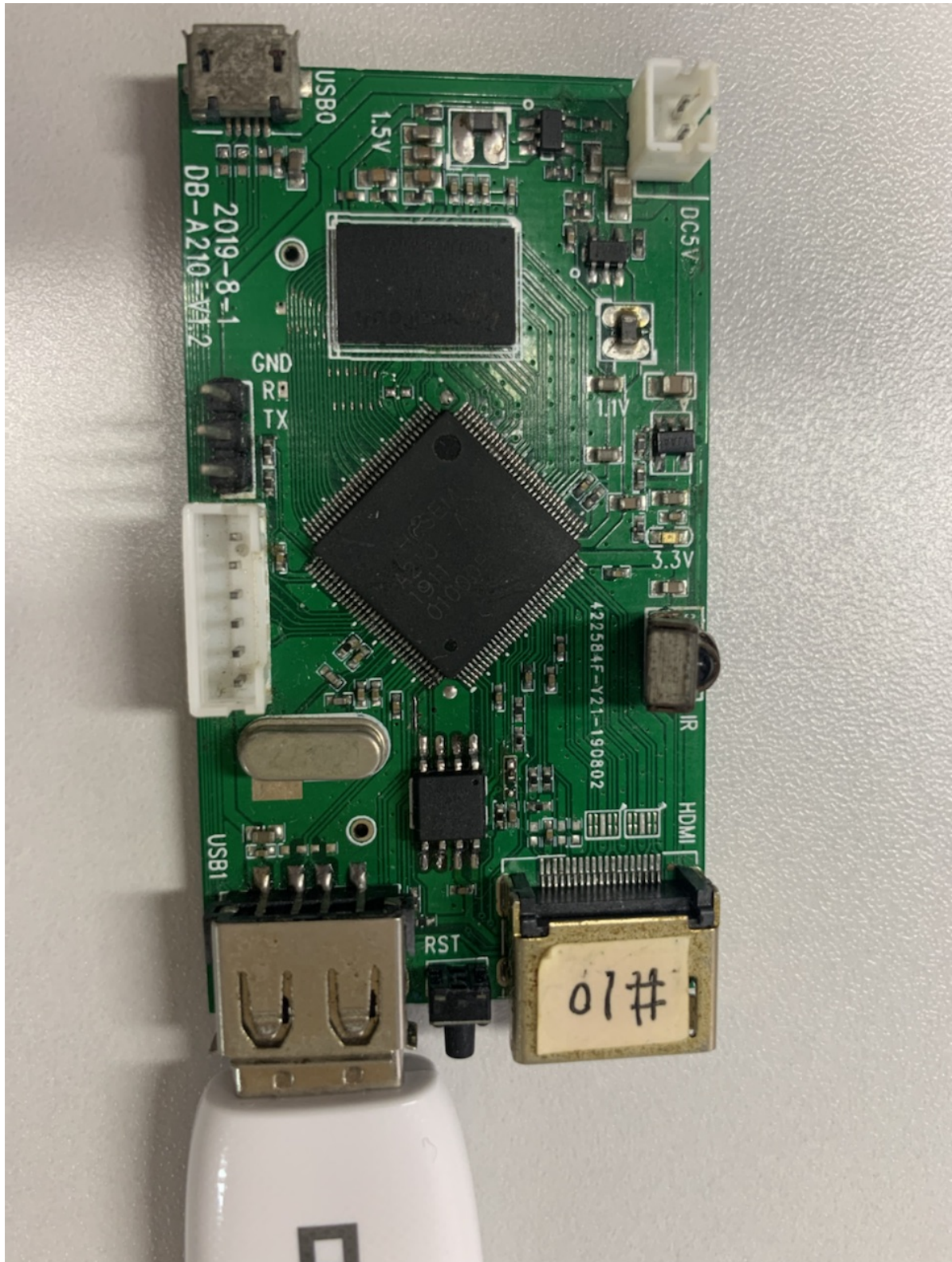
关于chip_number:

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

12.10.1 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_b1_defconfig
hichip_hc15xx_db_a210_v12_hcdemo_defconfig
```






2. 修改 hichip_hc15xx_db_a210_v12_hcdemo_b1_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=output-bl hichip_hc15xx_db_a210_v12_hcdemo_b1_defconfig
make O=output-bl all
make hichip_hc15xx_db_a210_v12_hcdemo_defconfig
make all
```

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

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

12.11 hcRTOS Nor Flash区间的分布

12.11.1 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";
};

sfsapi {
    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
hdmi
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)#

```

12.11.2 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:

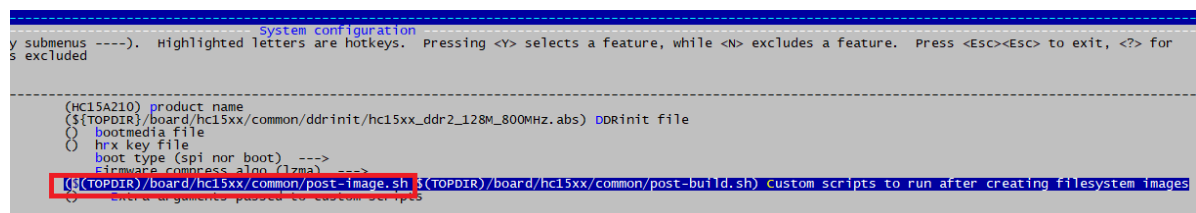
主要修改如下, 代码编译阶段会通过post-image.sh在image目录下生成一个预置的user_data的文件夹, 并格式化成user_data.bin文件。该文件就是放置于上述partition中的文件。

```

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:



```

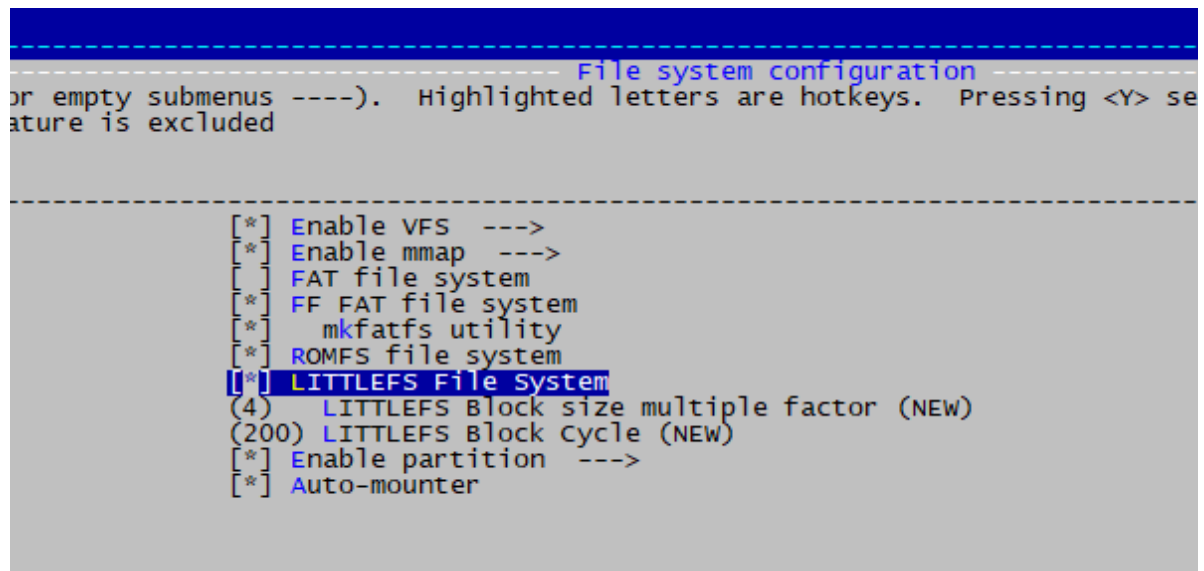
System configuration
-----
y submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> excludes a feature. Press <Esc><Esc> to exit, <?> for
s excluded

(HC15A210) product name
({TOPDIR}/board/hc15xx/common/ddrinit/hc15xx_ddr2_128M_800MHz.abs) DDRinit file
(O) bootmedia file
(O) hrx key file
boot type (spi nor boot) ---->
firmware compress algo (lzma) ---->
(H(TOPDIR)/board/hc15xx/common/post-image.sh $(TOPDIR)/board/hc15xx/common/post-build.sh) Custom scripts to run after creating filesystem images
(O) extra arguments passed to custom scripts

```

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=output-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;

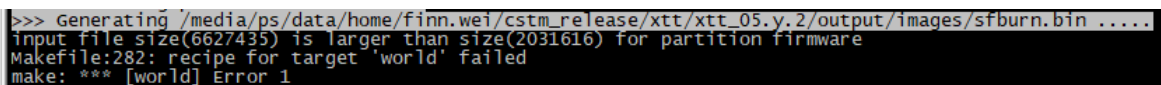
```

12.11.3 如何临时添加一个不支持的flash

参考：《HCCHIP SDK添加flash及修改速度说明-对外发布.pdf》

12.11.4 如何修改分区大小？

如果编译过程中遇到类似如下的错误：



```

>>> Generating /media/ps/data/home/finn.wei/cstm_release/xtt/xtt_05.y.2/output/images/sfburn.bin .....
input file size(6627435) is larger than size(2031616) for partition firmware
Makefile:282: recipe for target 'world' failed
make: *** [world] Error 1

```

需要注意该编译错误的以下3点：

1. 里面提到是 partition “firmware” 太小了。
1. 实际大小 (input file size) = 6627435 = 0x65206b。
1. 设置大小 (size) = 2031616 = 0x1f0000。

所以，我们要修改dts中：

```

sfspi {
    pinmux-active = <PINPAD_T14 1 PINPAD_T15 1 PINPAD_T16 1 PINPAD_T17 1
PINPAD_T18 1 PINPAD_T19 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 = <5>;

```



```

/* part0 is for entire norflash by default */

part1-label = "boot";                // 分区名字
part1-reg = <0x0 0x7f000>;           // 0x0为起始地址，
0x7f000为offset
part1-filename = "bootloader.bin";   // 分区文件

part2-label = "individual";
part2-reg = <0x7f000 0x1000>;         // 上一个分区的地址 +
offset为这个分区的 起始地址。以此类推
part2-filename = "hrxkey.bin";

part3-label = "eromfs";
part3-reg = <0x80000 0x80000>;
part3-filename = "romfs.img";

part4-label = "firmware";
part4-reg = <0x100000 0x6f0000>;      // 如上错误，那么把
firmware的分区大小改为0x6f0000， 大于0x65206b即可。
part4-filename = "projector.uImage";

part5-label = "persistentmem";
part5-reg = <0x7f0000 0x10000>;      // 对应的这个分区也需
要修改。 即 起始地址 = 上个分区起始地址 + 上个分区offset
part5-filename = "persistentmem.bin";
};
};

```

注意!!!

修改完一定需要重新编译：

```

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

```

12.11.5 如何读写flash里的OTP区域

参考：《HCRTOS flash的otp的读写说明文档.pdf》

12.12 uart蓝牙的配置

12.12.1 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：

dts部分

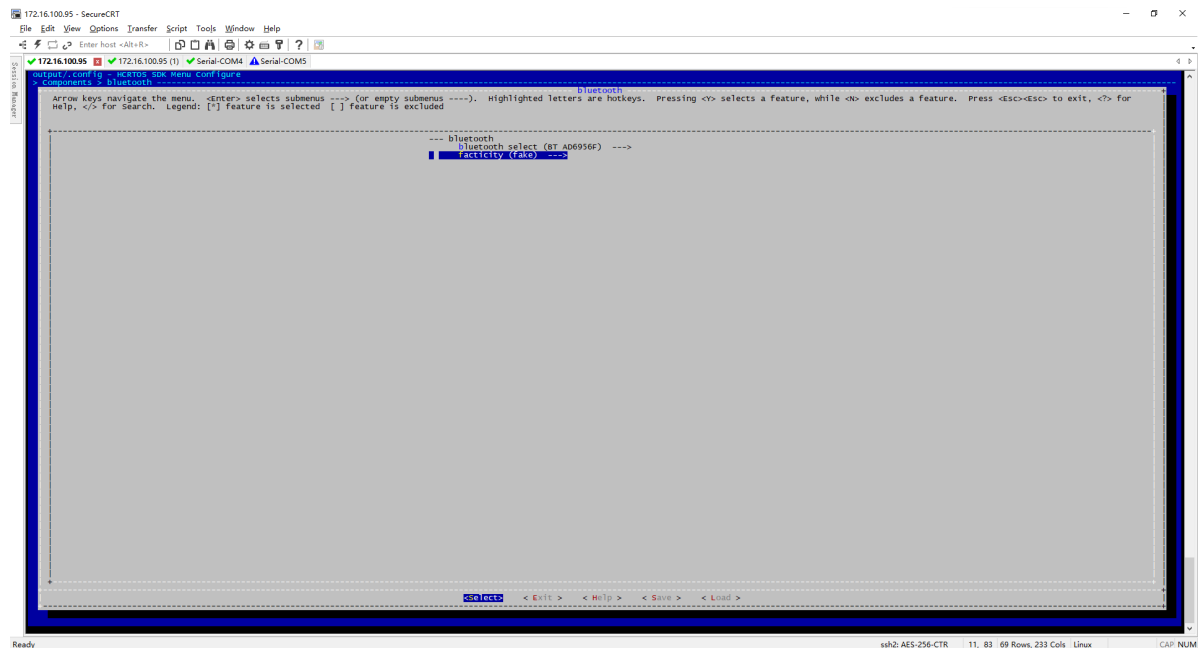
```
bluetooth {
    devpath = "/dev/uart1";//蓝牙硬件所连接的串口
    status = "disabled";//使能
};
```

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：



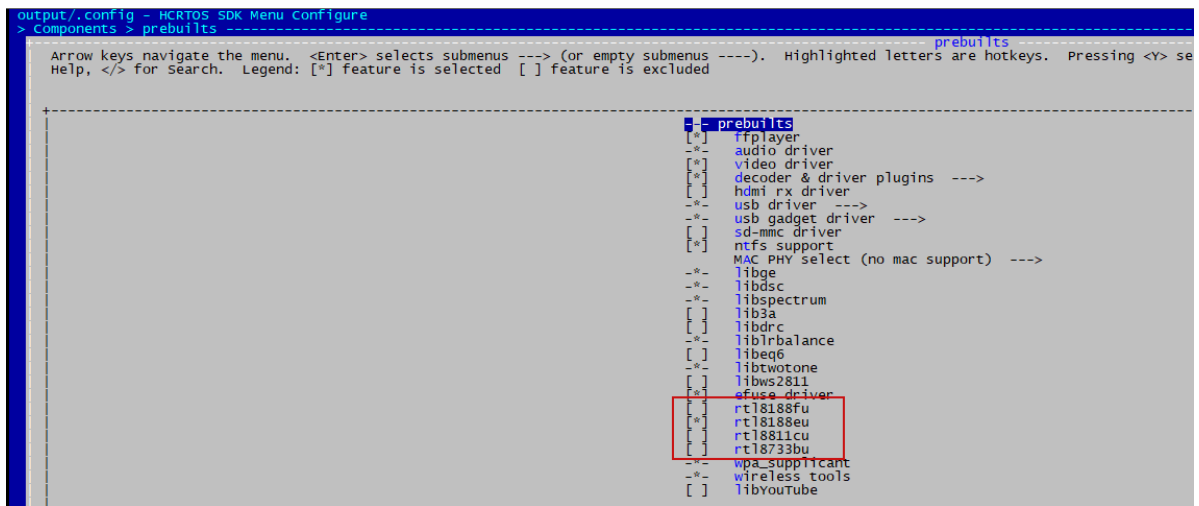
12.13 wifi的支持

目前hichip freeRTOS已经支持的wifi型号有：RTL8188FU、RTL8188EU、RTL8811CU、RTL8733BU。

```
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/rtl8733bu          //wifi 驱动，目前以库的形式提供
├─ Kconfig
└─ rtl8733bu.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《hcartos_wifi_user_guide.pdf》和《wifi_iperf测速使用说明文档.pdf》

配置wifi可以按如下图进行，根据板子上实际的wifi型号，选择相应的驱动：



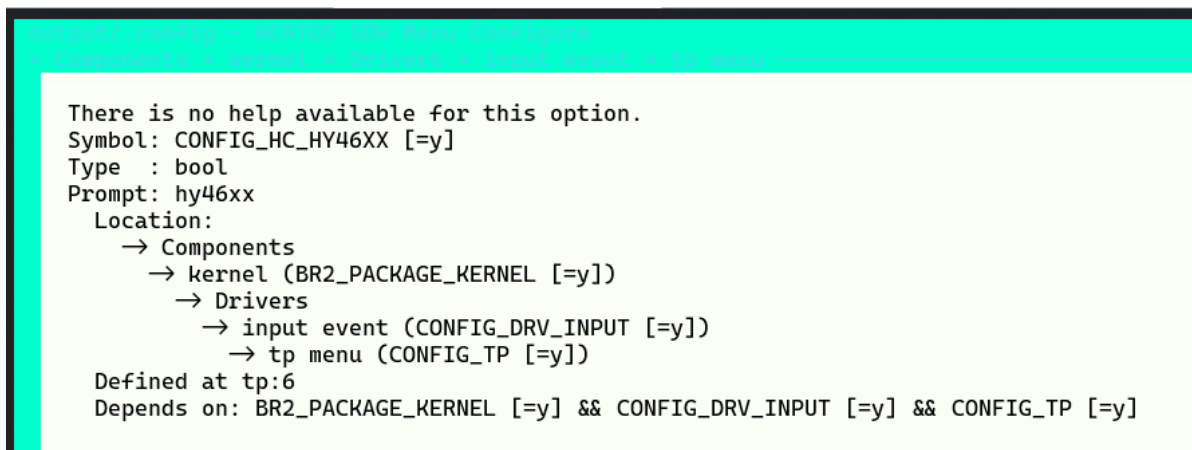
12.14 TP驱动的集成和测试

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

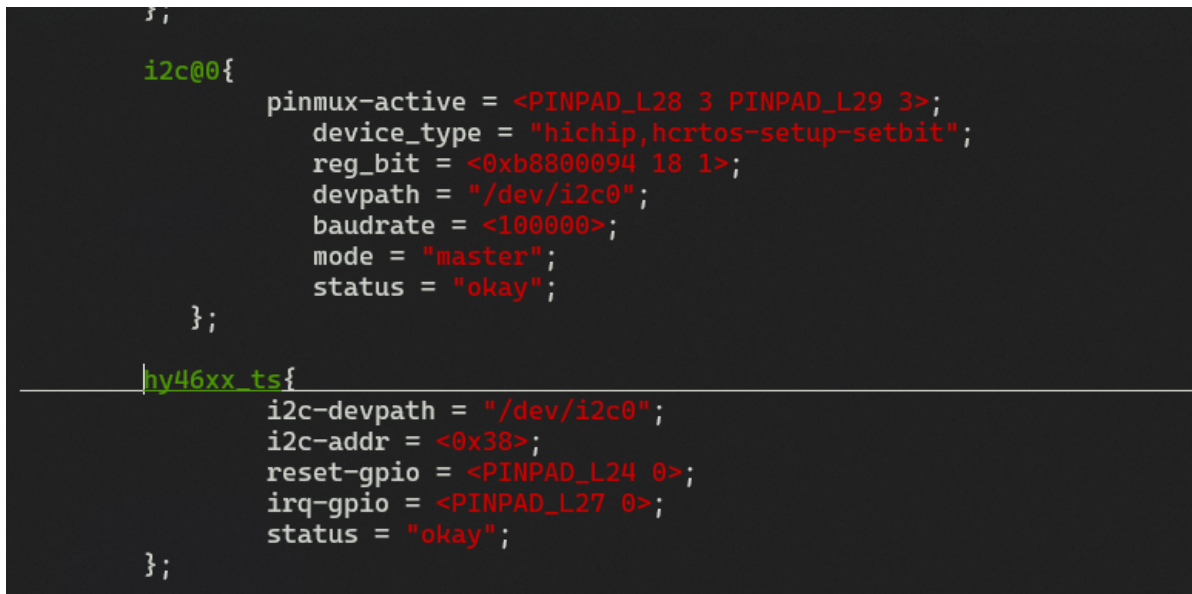
测试代码：\components\cmds\source\input_event

以hy46xx为例，需要的配置如下：

1. 打开驱动



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。



3. 测试代码的打开:

```
output/.config - HCRTOS 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=output-bl kernel-rebuild all
make kernel-rebuild hc-examples-rebuild cmds-rebuild all
```

触摸屏请参考以下文档:

《HCRTOS_btl1680触摸屏使用说明文档.pdf》

《HCRTOS xpt2046触摸屏使用说明文档.pdf》

《HCRTOS ILITEK_V5912触摸屏使用说明文档.pdf》

《HCRTOS hy46xx_ts触摸屏使用说明文档.pdf》

《HCRTOS GT911触摸屏使用说明文档.pdf》

《HCRTOS btl1680触摸屏使用说明文档.pdf》

12.15 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
```

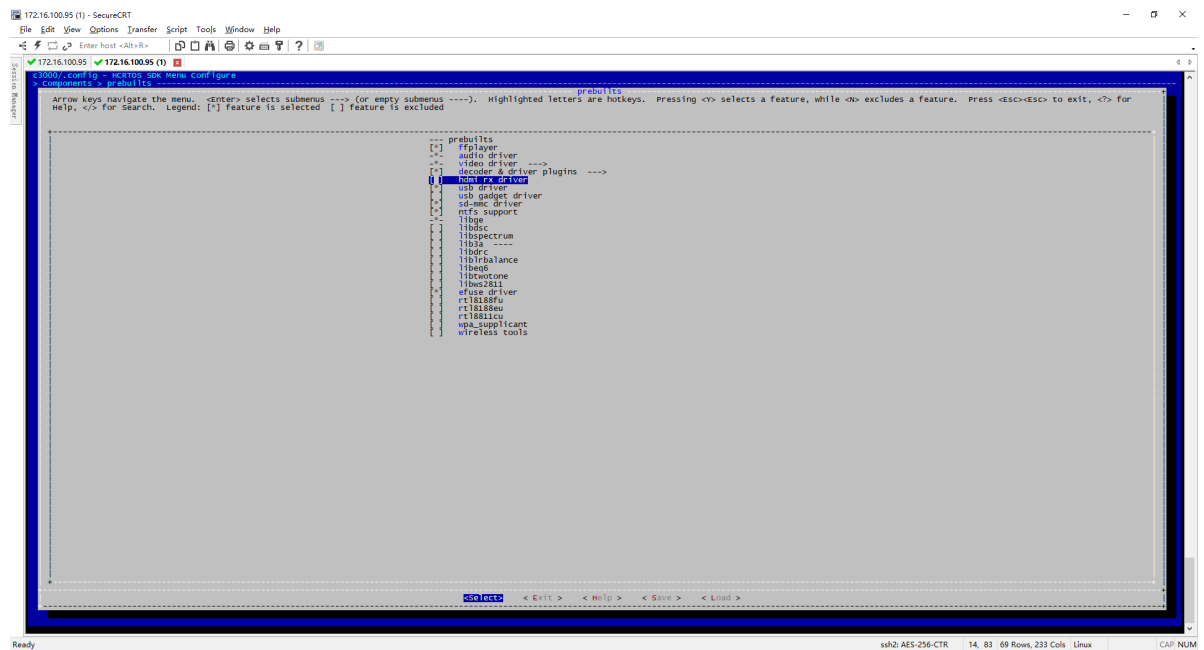
选上hdmi rx driver

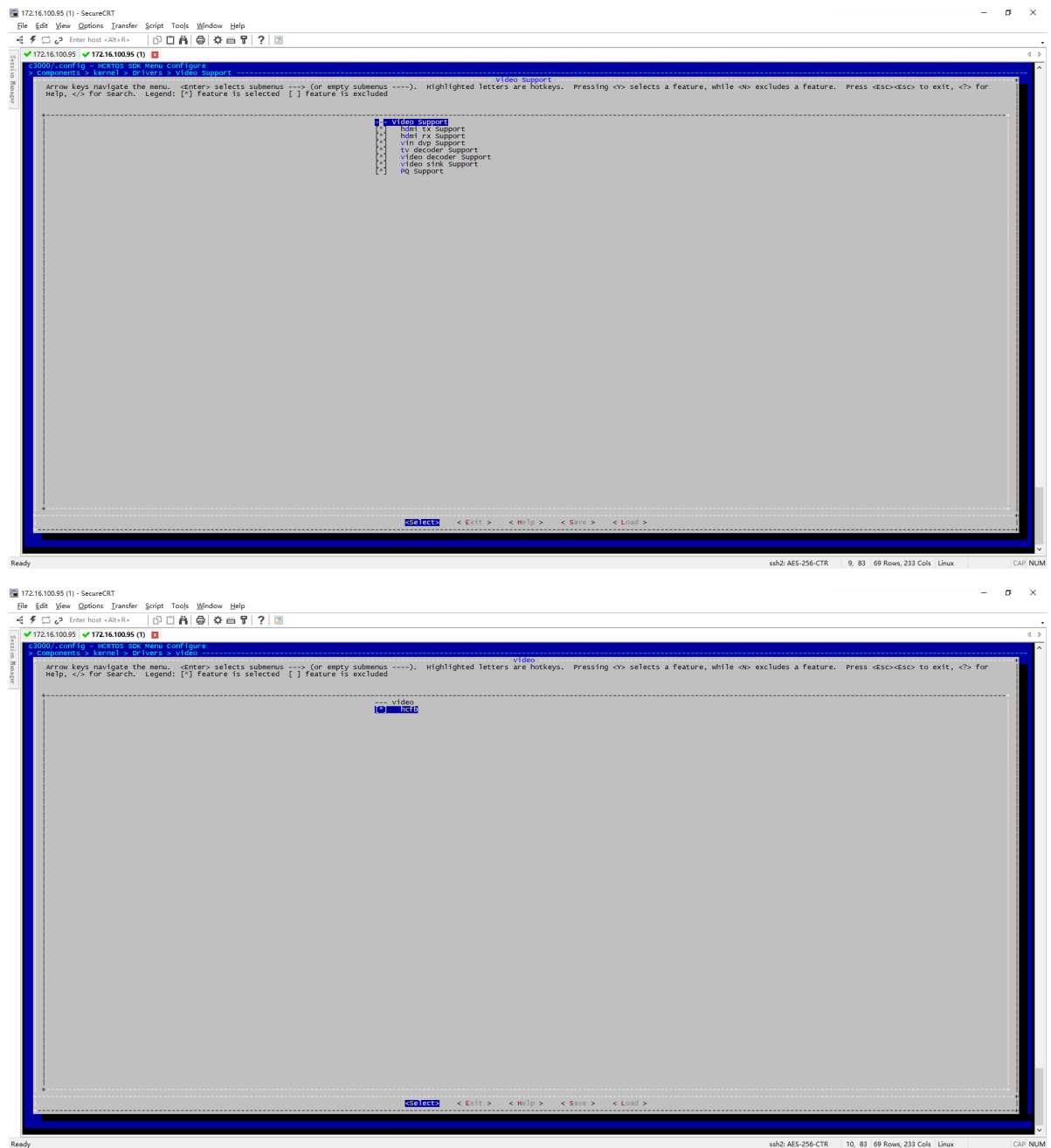
```
Components---> kernel---> Drivers---> Video Support---> hdmi rx Support
```

选上fb

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

退出并保存配置



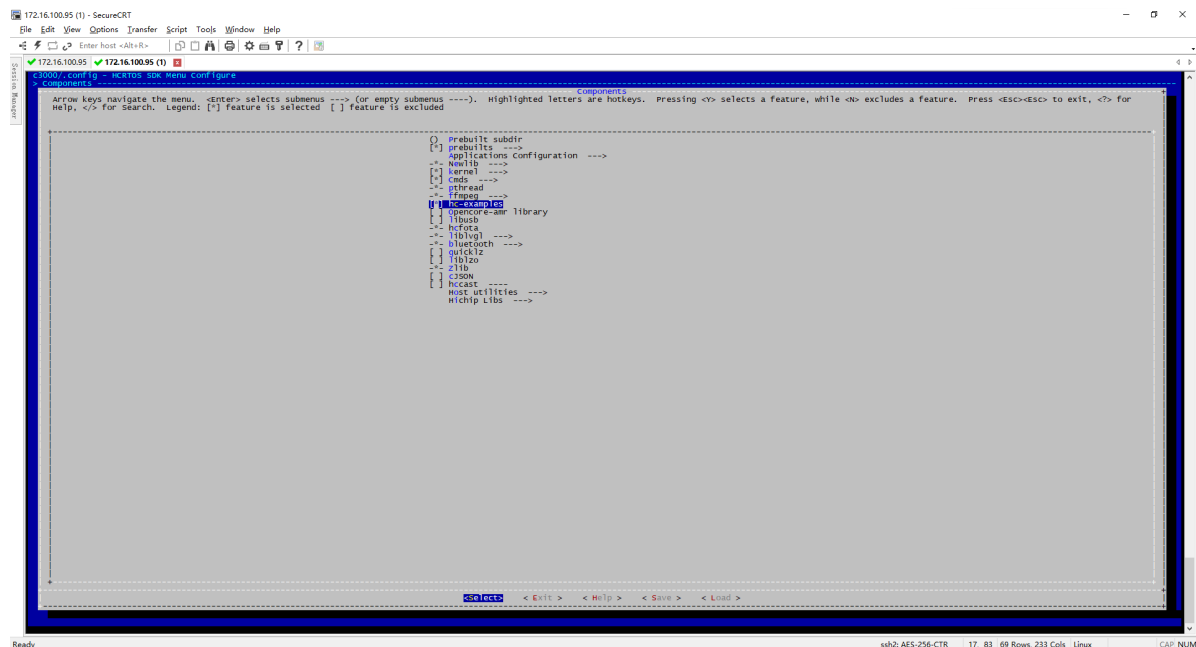


(2-2). 使能测试代码

```
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hdm i n

编译之前，退出并保存好配置

```
make O=output-bl 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。

12.16 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=output-bl kernel-rebuild all
make kernel-rebuild all

```

12.17 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

12.18 按键类支持

12.18.1 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-c1.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：

用户码: 00FF			
CMD:			
08			1C
55	5E	52	51
54	16		15
50	12		11
4C	0E		0D
49	0C		0A
4B	03		14
4F	5D		13
09	1A		05
47	06		07
18	48		17
56	57	1F	5B
43	41	10	4A
44	42	5A	4D
58	01	5F	19

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

gpio当作key的可以参考《HCRTOS key-gpio userguide.pdf》

12.19 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\dtb\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=output-bl kernel-rebuild all
make kernel-rebuild all
```

12.20 固件升级

12.20.1 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

12.20.2 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。

12.20.3 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

12.20.4 hcfota相关代码介绍

代码

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

12.20.5 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

12.20.6 hcfota调试：app如何调用hcfota接口

hftota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`。

串口升级可以参考《hcrtos_uart_upgrade_user_guide.pdf》

网络升级可以参考《hcrtos_net_upgrade_guide.pdf》

hcscreen的网络升级参考：《hcfota-hcscreen-network-upgrade-user-guide.pdf》

12.21 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:

12.21.1 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。

12.21.2 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。

12.21.3 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

12.21.4 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
    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

```

// 确认I2C配置是对的，有声音出来。

12.21.5 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。

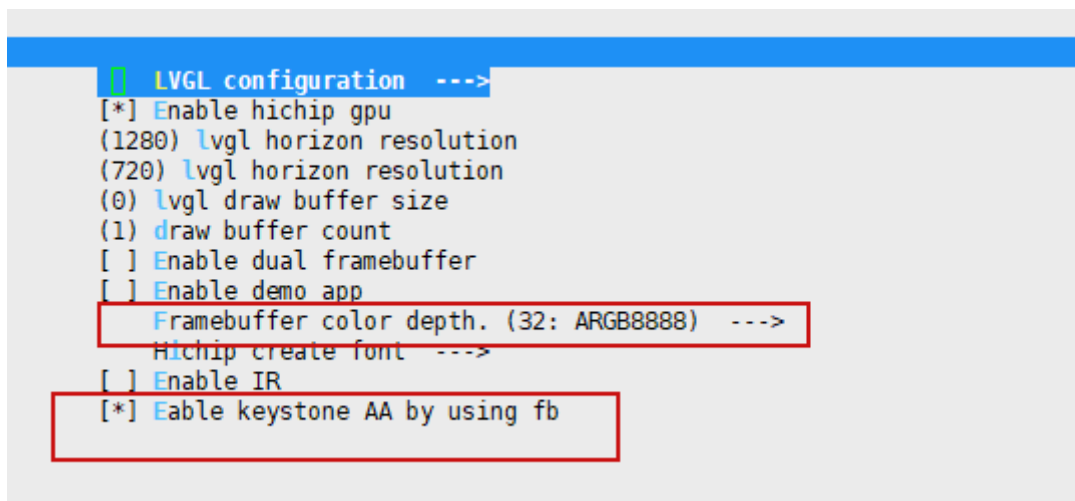
wm8960，也需要挂在I2C总线上，与cjc8988类似。

12.22 梯形校正

目前梯形校正只有在带屏的二代上使用，默认projector的app中可以设置。适用条件：横屏+0/180旋转+32bit UI；其他的芯片不支持。

梯形校正抗锯齿的使能，可以按如下配置：

make liblvgl-menuconfig



配置保存后，需要make liblvgl-rebuild all

12.23 boot standby 配置说明

步骤一 dts 上设置

```

standby {
    ir = <0xa8 0x1c 0x118>;    // ir key val
    adc = <0 0 100>;           //adc: 通道0 最小电压值 0mV 最大电压值 100mV
};

```

步骤二 boot standby的配置

```

1、选中 CONFIG_BOOT_STANDBY
cd hcartos
make O=output-bl menuconfig

```

```

Symbol: CONFIG_BOOT_STANDBY [=y]
-> Components
  -> Applications Configuration
    -> Application Selection
      -> bootloader (BR2_PACKAGE_APPS_BOOTLOADER [=y])

--- bootloader
[*]   Support boot standby

```

2、选中 CONFIG_BOOT_STANDBY 之后，默认会勾选，CONFIG_DRV_INPUT，CONFIG_DRV_STANDBY，如果没有请检查

3、检查 CONFIG_DRV_STANDBY 是否被勾选，该选项的内容是选中standby的驱动，没有该选项standby不会起作用（必要）

```

Location:
  -> Components
    -> kernel (BR2_PACKAGE_KERNEL [=y])
      -> Drivers
    *- standby driver --->

```

4、检查 CONFIG_DRV_INPUT 是否被勾选，该选项的内容是选中红外遥控器的驱动，没有该选项无法使用红外遥控器启动（可要）

```

-> Components
  -> kernel (BR2_PACKAGE_KERNEL [=y])
    -> Drivers
  *- input event --->

--- input event
[*]   rc core --->

```

boot standby 的代码位置，进入standby之前会关闭背光之类的内容，可以进行查看
components/applications/apps-bootloader/source/cmd/standby.c

步骤三 重新编译

```

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

```

12.24 如何添加adc key支持

参考《HCCHIP ADC开发文档.pdf》。

12.25 pin脚的功能查询

1. 查看芯片对应的datasheet，里面会有pin脚功能的定义。
2. 看完datasheet，继续看代码：hc16xx_pinmux.h或者hc15xx_pinmux.h，里面有对应的pin脚定义：

```
#define PINMUX_L13_GPIO          PINMUX_FUNCTION_SEL0
#define PINMUX_L13_UART0_RX      PINMUX_FUNCTION_SEL1
#define PINMUX_L13_I2C1_SCL      PINMUX_FUNCTION_SEL2
#define PINMUX_L13_HRX_SCL       PINMUX_FUNCTION_SEL3
#define PINMUX_L13_UART3_RX      PINMUX_FUNCTION_SEL4
#define PINMUX_L13_I2C0_SCL       PINMUX_FUNCTION_SEL5
#define PINMUX_L13_RGBHV_V_OUT   PINMUX_FUNCTION_SEL6
#define PINMUX_L13_UART2_RX      PINMUX_FUNCTION_SEL7
#define PINMUX_L13_I2C3_SCL       PINMUX_FUNCTION_SEL8
```

如上图定义，L13要配置成RX，那么对应的在dts中配置就是：

```
pinmux-active = <PINPAD_L13 1>;
```

如代码与datasheet不同，请以代码为准，并麻烦帮忙告知海奇窗口。

参考文档：《hertos-pinmux配置说明.pdf》

12.26 串口读写sample

参考《HCRTOS SDK uart userguide.pdf》

12.27 配置GPIO，使其在bootloader阶段就生效。

可以在dts中配置如下节点，保存后需要编译bootloader和app：make O=output-bl kernel-rebuild all; make kernel-rebuild all

```
gpio-out-def {
    gpio-group = <PINPAD_R03 GPIO_ACTIVE_LOW PINPAD_R04 GPIO_ACTIVE_LOW
PINPAD_L16 GPIO_ACTIVE_LOW PINPAD_L00 GPIO_ACTIVE_LOW>;
    status = "okay";
};
```

12.28 配置I2C sample

参考文档《HCRTOS i2c userguide.pdf》

12.29 如何修改edid进行测试

在 components/applications/apps-

projector/source/hcprojector_app/channel/hdmi_in/hdmi_rx.c 中添加需要设置的edid，比如：

```

static uint8_t edid_data[]=
{
    #if 1 //2488*720
        0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x21, 0x94, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00,
        0x0D, 0x1A, 0x01, 0x03, 0xA3, 0x79, 0x44, 0x96, 0x06, 0xEE, 0x91, 0xA3, 0x54,
        0x4C, 0x99, 0x26,
        0x0F, 0x50, 0x54, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
        0x01, 0x01,
        0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x02, 0x3A, 0xB8, 0x68, 0x91, 0xD0, 0x38,
        0x20, 0xA0, 0x20,
        0x46, 0x04, 0x26, 0xA5, 0x10, 0x00, 0x00, 0x1E, 0x13, 0x39, 0x80, 0x18, 0x71,
        0x38, 0x20, 0x40,
        0x30, 0x20, 0x88, 0x00, 0x26, 0xA5, 0x10, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00,
        0xFC, 0x00, 0x57,
        0x61, 0x76, 0x65, 0x53, 0x68, 0x61, 0x72, 0x65, 0x0A, 0x20, 0x20, 0x20, 0x00,
        0x00, 0x00, 0x00,
        0x00, 0x18, 0x78, 0x0C, 0x9A, 0x22, 0x00, 0x0A, 0x20, 0x20, 0x20, 0x20, 0x20,
        0x20, 0x01, 0x01,
        0x02, 0x03, 0x27, 0xC1, 0x50, 0x00, 0x05, 0x04, 0x03, 0x02, 0x07, 0x16, 0x01,
        0x1F, 0x12, 0x13,
        0x14, 0x20, 0x15, 0x11, 0x06, 0x23, 0x09, 0x07, 0x07, 0x65, 0x03, 0x0C, 0x00,
        0x10, 0x00, 0x83,
        0x01, 0x00, 0x00, 0xE3, 0x05, 0x03, 0x01, 0x02, 0x3A, 0x80, 0x18, 0x71, 0x38,
        0x2D, 0x40, 0x58,
        0x2C, 0x45, 0x00, 0x06, 0x44, 0x21, 0x00, 0x00, 0x1E, 0x01, 0x1D, 0x80, 0x18,
        0x71, 0x1C, 0x16,
        0x20, 0x58, 0x2C, 0x25, 0x00, 0x06, 0x44, 0x21, 0x00, 0x00, 0x9E, 0x01, 0x1D,
        0x00, 0x72, 0x51,
        0xD0, 0x1E, 0x20, 0x6E, 0x28, 0x55, 0x00, 0x06, 0x44, 0x21, 0x00, 0x00, 0x1E,
        0x8C, 0x0A, 0xD0,
        0x8A, 0x20, 0xE0, 0x2D, 0x10, 0x10, 0x3E, 0x96, 0x00, 0x06, 0x44, 0x21, 0x00,
        0x00, 0x18, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x67
    #endif
};
hdmi_rx_edid_data_t hdmi_rx_edid;

```

再在 `hdmirx_start` 函数中，设置 HDCP KEY 前做如下调用：

```

memcpy(&(hdmi_rx_edid.edid_data[0]), edid_data, EDID_DATA_LEN);
ioctl(g_switch_rx.in.fd, HDMI_RX_SET_EDID , &hdmi_rx_edid);
printf("test begin set edid .....\\n");

```

重编译后系统起来就是需要的 edid 设置了。

!!! 注意：这种方法仅限测试开发用，生成请用正规方法。

12.30 PQ 工具使用说明

参考《1600 hcRTOS PQ 工具使用说明.pdf》

12.31 如何使用 RTC?

参考《HCRTOS rtc_at8563 使用说明文档.pdf》

12.32 USB gadget相关驱动说明

参考《hcRTOS USB gadget mass-storage 驱动说明.pdf》/《hcRTOS USB gadget NCM 驱动说明.pdf》/《hcRTOS USB gadget Zero 驱动说明.pdf》/《hcRTOS USB HID 驱动说明.md》/《hcrtos winusb(wcid)驱动使用说明.pdf》

12.33 watch dog相关说明

参考《HCRTOS watchdog timer userguide.pdf》

12.34 PHY相关说明

参考《hcRTOS 裕太phy驱动使用说明.pdf》

12.35 如何在SDK中集成第三方库？

参考《HCRTOS-third-party-lib-integration-V2.pdf》

12.36 gsensor的集成和使用

参考《HICHIP SC7A20 gsensor userguide.pdf》

12.37 hcrtos死机时调试小方法

参考《stack_probe_tool_for_debug.pdf》

12.38 投影仪声音调节的方法

参考《投影仪声音调节指导说明.pdf》

12.39 配屏

请参考《hcrtos_lcd_user_guide.pdf》和《HCRTOS pwm userguide.pdf》

12.40 图片解码的规格

请参考《图像解码规格.pdf》

12.41 airp2p配置常见问题

demo软件中，airp2p需要的配置如下：

```
CONFIG_SCHED_HRTWORK=y
CONFIG_SCHED_HRTNTHREADS=1
CONFIG_SCHED_HRTWORKPRIORITY=16
CONFIG_SCHED_HRTWORKSTACKSIZE=0x1000
BR2_PACKAGE_HCCAST_AIRP2P_SUPPORT=y
CONFIG_APPS_PROJECTOR_AIRP2P=y
```

另外，airp2p不支持2.4G的wifi。

12.42 如何离线更换image里的logo文件？

gitlab路径：\tools\固件烧写\离线更换image里的logo的方法.pdf

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 --force
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=output-b1 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 -lvldrv_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
```

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请联系海奇窗口。

13.7 发现苹果同屏不可用

请接上打印看打印信息，并提供打印信息给原厂。

13.8 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";
};
```

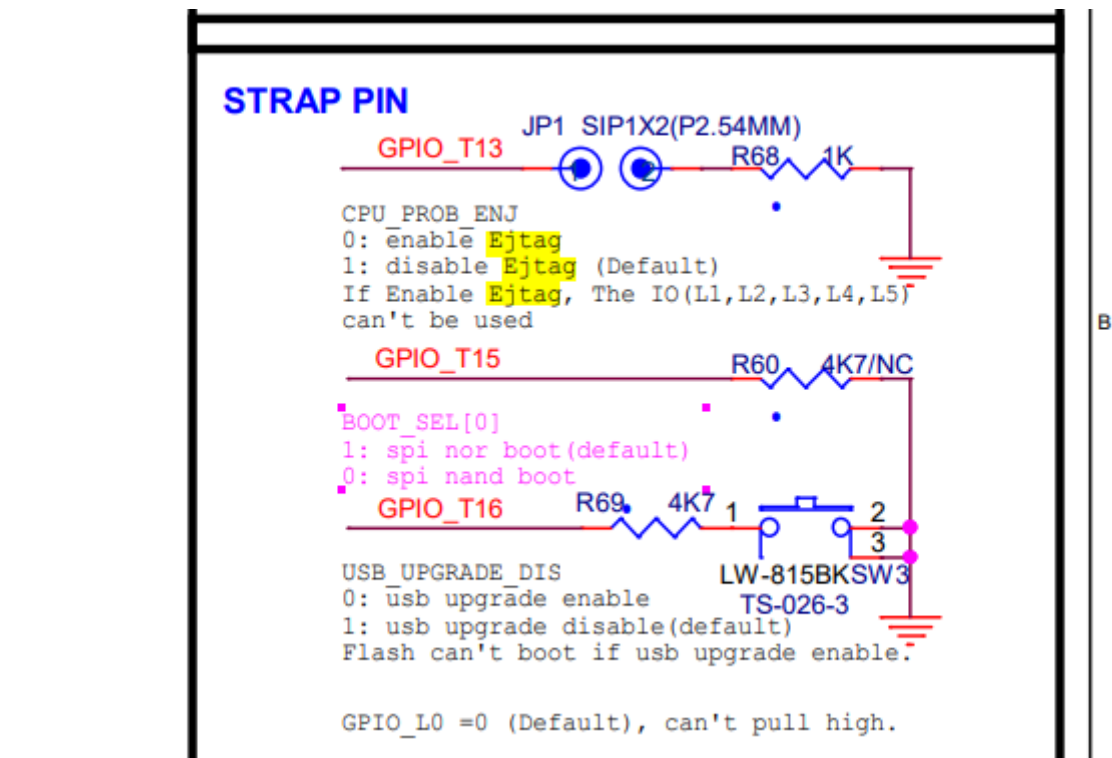
打开后重新编译：

```
make O=output-bl 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。



13.9 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：

[海奇【TASK/BUG管理系统】\(hichiptech.com\)](http://hichiptech.com)

原厂工程师看到后会第一时间进行处理。

报问题原则：

1. 碰到问题优先在document和gitlab上找文档。
2. 如无文档，请参考代码。

3. 如代码还无法解决，请联系支持窗口，可通过微信与电话与窗口联系。
4. 如还无法解决，请按redmine方式上报问题。

redmine上报问题原则：

1. 标题：填写 芯片型号+项目+问题描述
2. 指派给：该项不要改，默认即可。
3. 客户硬件环境：填写上复现问题的硬件环境，最好能用demo先试下，对比demo是否可以直接复现。
4. 报问题的SDK版本号：填写你目前复现问题的SDK版本。
5. 问题详细描述：描述操作环境、复现方法、所需文件、最好有复现的视频介绍。

如不按以上规则报问题，可能会增加沟通成本，导致问题无法及时解决。

13.10 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。

13.11 如何增加 简易的 测试api 或者 复现问题 的代码？

参考\components\cmds下的例子。

可以在menuconfig里选择，选完后重编代码：`make kernel-rebuild cmds-rebuild all`

之后即可在串口中输入命令。

13.12 如何打patch？

在SDK的开发过程中，原厂会解问题，随之会将patch更新给开发者，这时可以有两种打patch的方法：

1. 直接对比代码，将差异文件更新到开发的版本中。

优点：可以直观的看出修改的文件，清楚把握代码的修改。

缺点：繁琐，容易遗漏。

2. 用git命令来打patch：`git apply --ignore-space-change --ignore-whitespace patch_file`

优点：省心。

缺点：不容易看出修改的点。

13.13 客户代码如何打包成库文件

参考 documents/海奇FreeRTOS SDK三方库集成方法-V2.pdf

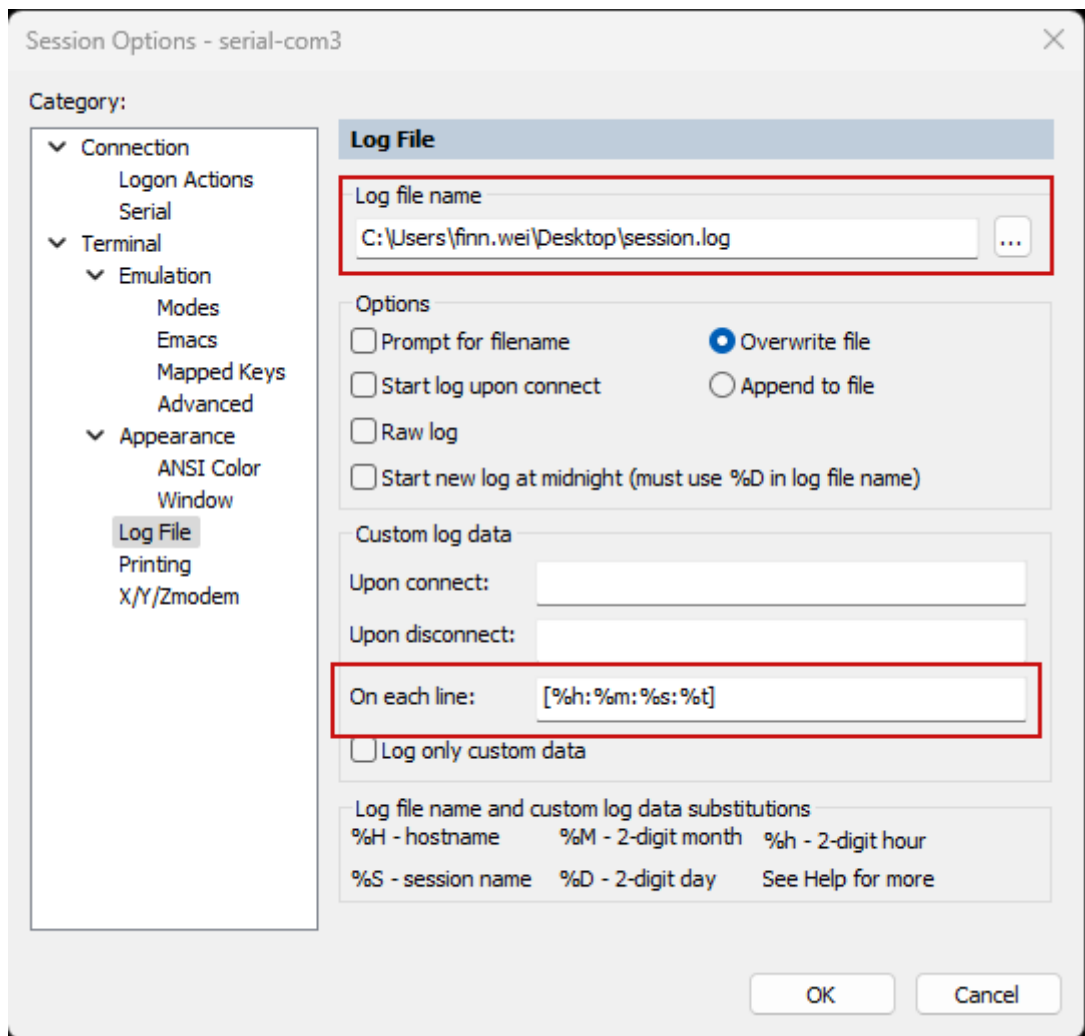
13.14 如何提升开机时间？

13.14.1 影响开机时间的因素主要有以下几点：

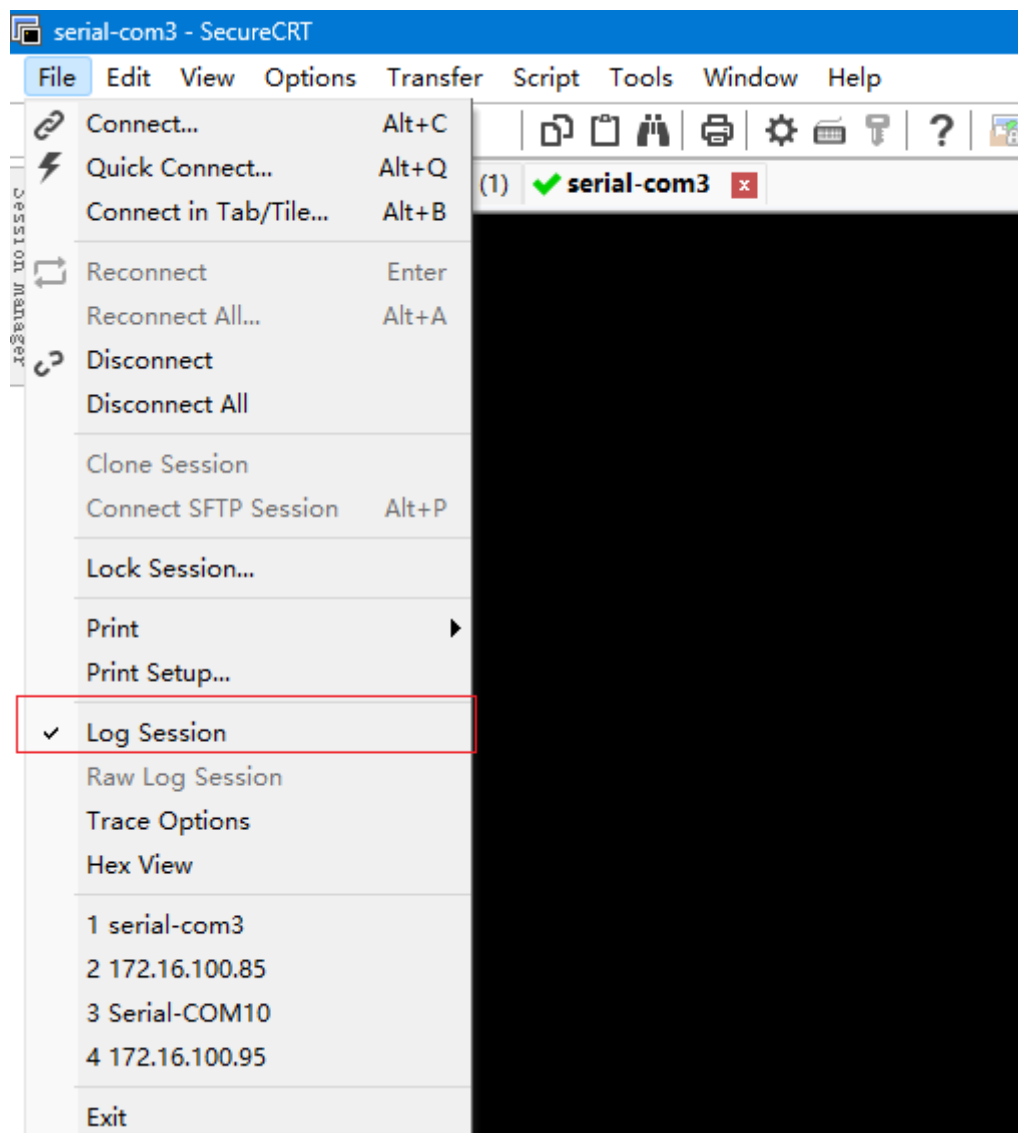
1. 软件功能太多
2. 最终软件太大
3. 压缩率太大。
4. flash读写速度不够快
5. UI显示慢
6. 没有show boot logo

13.14.2 主要调试手段：

1. 需要接上打印，使用SecureCRT工具，将打印的时间戳打印出来：



2. 保存好log:



3. 这样接好串口，再上电开机，就可以在session.log中看到开机的详细打印时间了。然后分析时间具体是消耗在哪个步骤的。有针对性的减少这些时间。

13.14.3 主要方法

1. 关闭boot中不必要的功能，可以在 `make O=output-bl menuconfig` 中进行配置。
2. 尽量使用低分辨率的UI素材等，flash中尽量保存少的图片、视频、音乐等大的素材。
3. 我们目前支持lzma/gzip/lzox1三种压缩方式，经过实测，lzma的压缩率最大，有利于小flash的应用，gzip介于两者之间，有利于快速开机需求。
4. 一代芯片可以使用flash双线模式。并且将频率设置成50M. 二代芯片可以考虑使用4线模式，但需要确认硬件和flash均支持。

```
sfspi {
    pinmux-active = <PINPAD_L11 1 PINPAD_L12 1 PINPAD_L13 1 PINPAD_L14
1>;
    sclk = <50000000>;          //从27000000改成50000000
    dma-mode = <1>;
    status = "okay";

    spi_nor_flash {
        spi-tx-bus-width = <2>;    //从1改成2
        spi-rx-bus-width = <2>;    //从1改成2
        reg = <0>;
        status = "okay";
    }
}
```

5. 尽量将UI的显示提前，lvgl初始化完即可。
6. 打开boot show logo，也是快速开机的一种方法。

13.15 reserve
