HOPERUN
Information Technology
潤和軟件

Akebi96 User's Guide

Issue: 2.0

Date: 2019-12-05

Trademarks and Permissions

and other HopeRun icons are trademarks of Jiangsu HopeRun Software Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HopeRun and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Jiangsu HopeRun Software Co., Ltd.

Address: HopeRun Software Park
Software road No.168, Yuhuatai
Nanjing 210012
People's Republic of China

Website: http://www.hoperun.com

Email: hihope@hoperun.com

About This Document

Purpose

This document describes how to build and flash images to a akebi96 board and test akebi96's dvb driver.

Related Version

The following table lists the product version related to this document.

Product Name	Version
Akebi96	all

Intended Audience

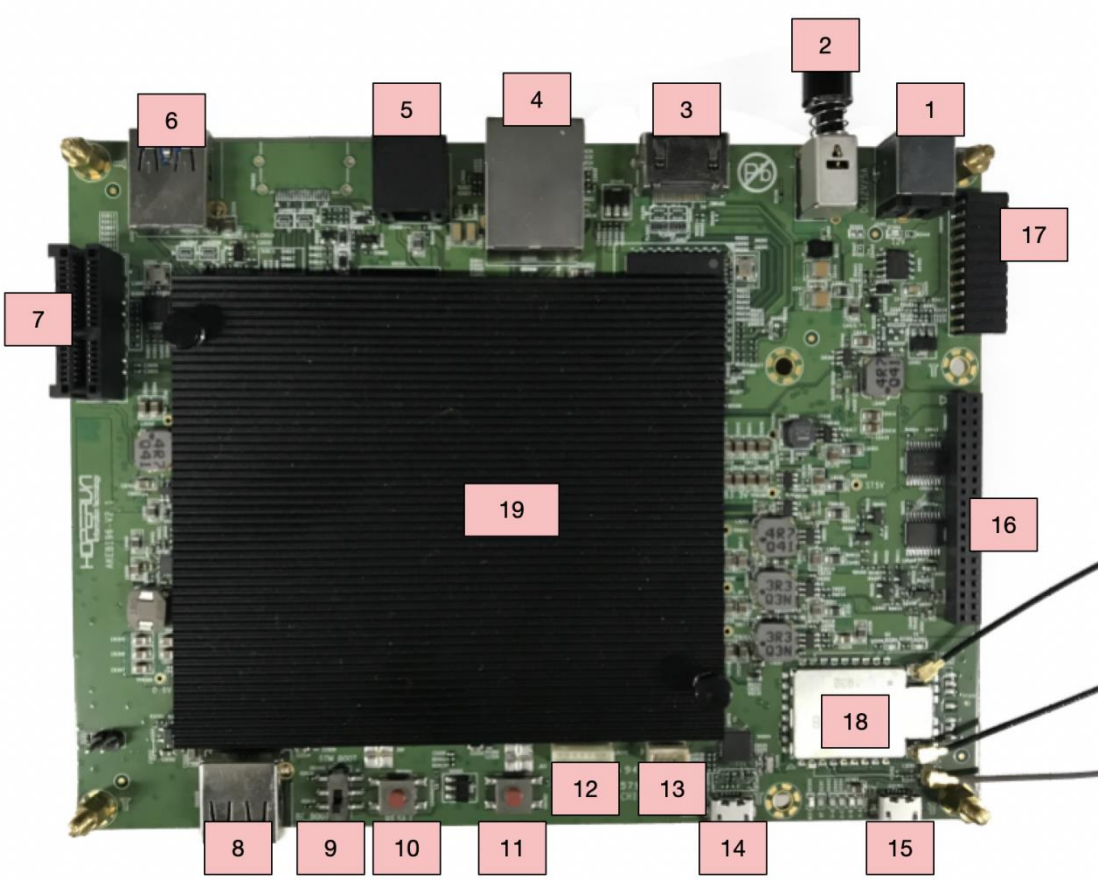
This document is intended for:

Technical support engineers

Software development engineers

Catalog

1. Introduction



1	DC Jack	2	Power Switch
3	HDMI TX Connector	4	Ether Connector
5	Coaxial/SPDIF Connector	6	USB 3.0 Port
7	8KDEC PCIE Connector	8	USB 2.0 Port
9	Boot Mode Switch	10	Reset Key
11	USB Boot key	12	MCU ICSP Connector
13	MCU UART Connector	14	USB Micro-B Port / UART1
15	USB Micro-B Port /UART0/ STM	16	Expansion Connector
17	Tuner Card IF Connector	18	BT/WIFI Module
19	SC1408A Processor		

2. Boot Board

(1). Download binary images

```
cd ~
git clone https://github.com/hihope-akebi96/binary-images.git
```

(2). Flash binary images to board

i. Prepare USB stick

Set up USB memory formatted with FAT32 on PC, and find device file from dmesg. (ex. /dev/sdc1)

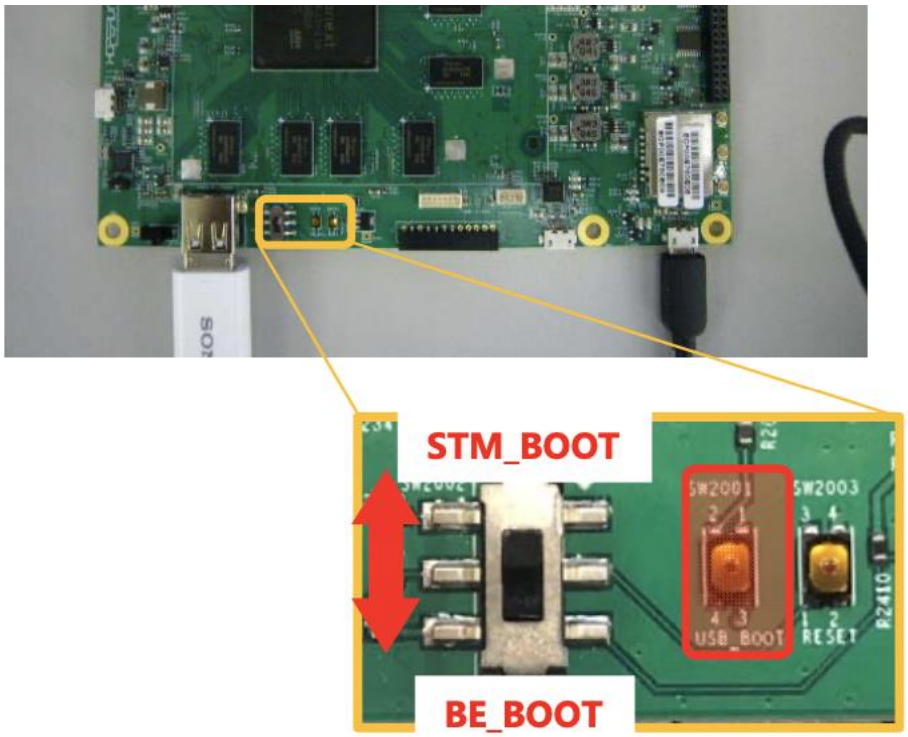
Copy all files under "~/binary-images" into "usb" folder in USB stick.

```
sudo mount /dev/sdc1 /mnt
sudo mkdir -p /mnt/usb
sudo cp ~/binary-images/*    /mnt/usb/
sudo umount /mnt
```

ii. Preparation of target board.

Boot mode must be set as "BE mode"

* Set as "BE BOOT" on "SW2002".



iii. Connect USB stick to board usb 3.0 port.

iv. Connect PC to board uart1 port with usb cable.

v. Push power switch tp power on board, and stop uboot prompt by any key on PC Serial Terminal(eg, minicom). After that, enter "run update_from_usb", then start to flash all rom images. It takes abount 90 second.

```
U-Boot 2019.01 (Feb 28 2019 - 14:31:04 +0900)

SoC:      LD20 (model 1, revision 2)
Model: Akebi96 Board
DRAM:    3 GiB
SC:        Micro Support Card (CPLD version 15.15)
NAND:    nand_base: timeout while waiting for chip to become ready
nand_base: No NAND device found
0 MiB
MMC:      sdhc@5a000000: 0
Loading Environment from MMC... OK
In:        serial@54006800
Out:      serial@54006800
Err:      serial@54006800
MODE:    eMMC Boot (STM: OFF)
Net:
Warning: ethernet@65000000 (eth0) using random MAC address - 3a:57:2d:50:4e:26
eth0: ethernet@65000000
Hit any key to stop autoboot:    0
=>
=> run update_from_usb
5269 bytes read in 26 ms (197.3 KiB/s)
## Executing script at 8c000000
**************************************
*** set GPT partition ...                    ***
**************************************
switch to partitions #0, OK
mmc0(part 0) is current device
Writing GPT: success!
**************************************
*** Writing to boot partition 0 ... ***
**************************************
switch to partitions #1, OK
mmc0(part 1) is current device
39705 bytes read in 31 ms (1.2 MiB/s)

MMC write: dev # 0, block # 0, count 256 ... 256 blocks written: OK
425472 bytes read in 23 ms (17.6 MiB/s)

MMC write: dev # 0, block # 256, count 2816 ... 2816 blocks written: OK
```

```
**************************************
*** Writing to normal area ...           ***
**************************************
switch to partitions #0, OK
mmc0(part 0) is current device
6610395 bytes read in 95 ms (66.4 MiB/s)

MMC write: dev # 0, block # 80, count 24576 ... 24576 blocks written: OK
23773 bytes read in 26 ms (892.6 KiB/s)

MMC write: dev # 0, block # 24656, count 64 ... 64 blocks written: OK
6899565 bytes read in 87 ms (75.6 MiB/s)

MMC write: dev # 0, block # 24720, count 131072 ... 131072 blocks written: OK
6612992 bytes read in 71 ms (88.8 MiB/s)

MMC write: dev # 0, block # 155792, count 16520 ... 16520 blocks written: OK
12673024 bytes read in 156 ms (77.5 MiB/s)

MMC write: dev # 0, block # 172312, count 40824 ... 40824 blocks written: OK
107151930 bytes read in 725 ms (140.9 MiB/s)
Uncompressed size: 268435456 = 0x10000000

MMC write: dev # 0, block # 213136, count 524288 ... 524288 blocks written: OK
126873443 bytes read in 1442 ms (83.9 MiB/s)
Uncompressed size: 268435456 = 0x10000000

MMC write: dev # 0, block # 737424, count 524288 ... 524288 blocks written: OK
93456661 bytes read in 1077 ms (82.8 MiB/s)
Uncompressed size: 268435456 = 0x10000000

MMC write: dev # 0, block # 1261712, count 524288 ... 524288 blocks written: OK
118157455 bytes read in 1355 ms (83.2 MiB/s)
Uncompressed size: 268435456 = 0x10000000

MMC write: dev # 0, block # 1786000, count 524288 ... 524288 blocks written: OK
42349274 bytes read in 309 ms (130.7 MiB/s)
Uncompressed size: 268435456 = 0x10000000

MMC write: dev # 0, block # 2310288, count 524288 ... 524288 blocks written: OK
260547 bytes read in 32 ms (7.8 MiB/s)
Uncompressed size: 268435456 = 0x10000000

MMC write: dev # 0, block # 2834576, count 524288 ... 524288 blocks written: OK
```

```
54509644 bytes read in 639 ms (81.4 MiB/s)

MMC Sparse write: dev # 0, block # 3358864 ... Flashing Sparse Image
........ wrote 54509568 bytes to '0x66812000'
1737068 bytes read in 42 ms (39.4 MiB/s)

MMC Sparse write: dev # 0, block # 3621008 ... Flashing Sparse Image
........ wrote 1736704 bytes to '0x6e812000'
319756 bytes read in 27 ms (11.3 MiB/s)

MMC Sparse write: dev # 0, block # 12286096 ... Flashing Sparse Image
........ wrote 319488 bytes to '0x176f12000'
**************************************
*** USB update is finished.                ***
**************************************
```

vi. Reset board.

```
=> reset
U-Boot 2019.01 (Feb 28 2019 - 14:31:04 +0900)

SoC:      LD20 (model 1, revision 2)
Model: Akebi96 Board
DRAM:    3 GiB
SC:        Micro Support Card (CPLD version 15.15)
NAND:    nand_base: timeout while waiting for chip to become ready
nand_base: No NAND device found
0 MiB
MMC:      sdhc@5a000000: 0
Loading Environment from MMC... OK
In:        serial@54006800
Out:      serial@54006800
Err:      serial@54006800
MODE:    eMMC Boot (STM: OFF)
Net:
Warning: ethernet@65000000 (eth0) using random MAC address - 3a:57:2d:50:4e:26
eth0: ethernet@65000000
Hit any key to stop autoboot:    0
=>
=> env default -a
## Resetting to default environment
=> saveenv
Saving Environment to MMC... Writing to MMC(0)... OK
```

After these operations, execute reset or power off -> on, then the board will be wake up by new

ROM.

3. Build Images

1.1. Install Packages

```
apt-get install --fix-missing -y git bc cmake ncurses-dev autoconf bison ccache cscope curl flex gdisk libfdt-dev libglib2.0-dev libpixman-1-dev netcat python-crypto python-serial uuid-dev xz-utils zlib1g-dev gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libsdl1.2-dev xterm cpio libssl-dev rsync

apt-get install -y mtd-utils genromfs sudo stgit device-tree-compiler python3 iputils-ping iasl sparse bsdmainutils u-boot-tools img2simg repo openjdk-8-jdk ccache libgl1-mesa-dev libxml2-utils xsltproc lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z1-dev zip dosfstools mtools simg2img connect-proxy locales python-mako python-pycryptopp kmod

cd /opt/
wget
http://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/arm-linux-gnueabihf/gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
tar xf gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
wget
http://releases.linaro.org/components/toolchain/binaries/7.3-2018.05/aarch64-linux-gnu/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
tar xf gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
export
PATH="$PATH:/opt/gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabihf/bin:/opt/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin"
```

1.2. Build bootload & kernel

(1). Downlaod bootload & kernel source codes

```
mkdir ~/aosp/bsp
cd ~/aosp/bsp
git clone https://github.com/buildroot/buildroot -b 2018.02.6 --single-branch
git clone https://github.com/96boards-akebi96/buildroot-configs.git -b master --single-branch
```

(2). Build bootload & kernel source codes

```
cd ~/aosp/bsp/buildroot
make BR2_EXTERNAL=../buildroot-configs akebi96_defconfig
make clean
make
```

Notes :

i. Do not add "-j" option. It will be optimised by "buildroot".

ii. Internet connection must be available on PC for using build.
    * Packages needed to be build will be download automatically at the first build.
    * After the first time, these will be cached under "bsp/build-local/dl" .

iii. For buildroot, please refer following links.
    * https://buildroot.org/
    * https://buildroot.org/downloads/manual/manual.html

1.3. Build android

(1). Download android source code

```
cd ~/aosp/android
repo init -u https://android.googlesource.com/platform/manifest -b android-9.0.0_r34
git clone -b sni-release --single-branch
https://github.com/96boards-akebi96/akebi96-manifests.git .repo/local_manifests
repo sync -j8
```

(2). Build android source code

```
cd ~/aosp/android
source build/envsetup.sh
lunch akebi96-userdebug
make -j8
./make_romimage.sh
```

Notes :

i. Android ROM will be installed in "~/aosp/bsp/buildroot/output/images" after these procedure.

ii. This may take a few hours.

1.4. Flash images

All builded images are installed in "~/aosp/bsp/buildroot/output/images". Flash them flow Chapter 2.

4. Test Tuner

4.1 Establish mini digital TV transmission base station.

In the test state, we use "DekTec output adapters" & "streamxpress player" to establish mini digital TV transmission base station in the laboratory.

(1). You need a “Dektec output adapter”.

List of supported devices:

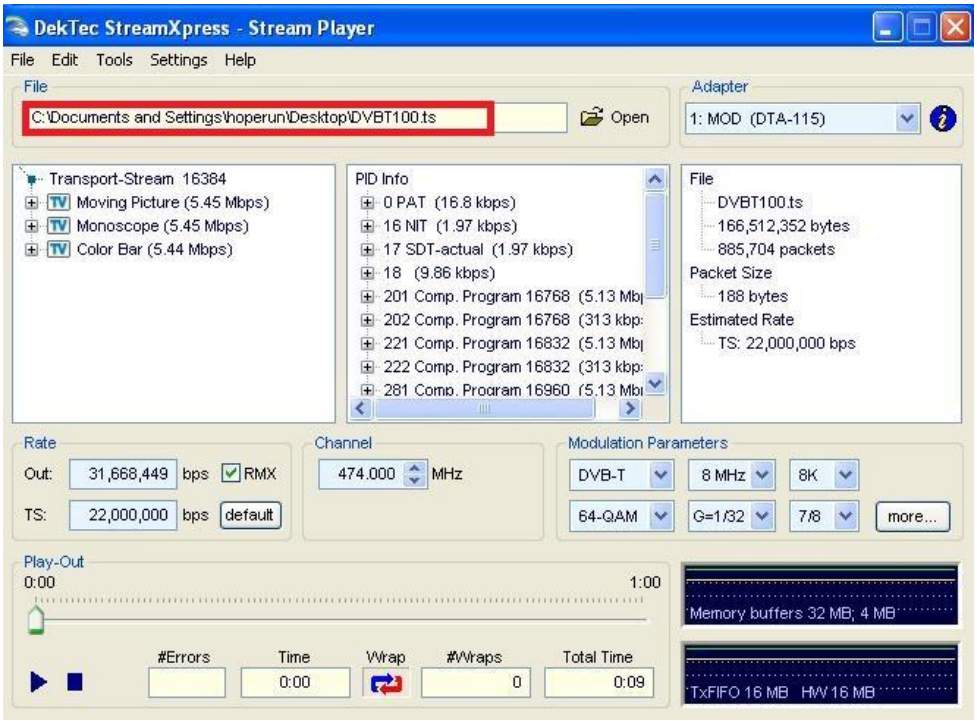
```
1. DTA-100，105，112，115
```

2. DTA-116，117，145，160
3. DTA-2136，2137C，2142
4. DTA-2144B，2145，2152
5. DTA-2154，2160，2172
6. DTA-2174，2175，2179
7. DTA-2195
8. DTE -3100

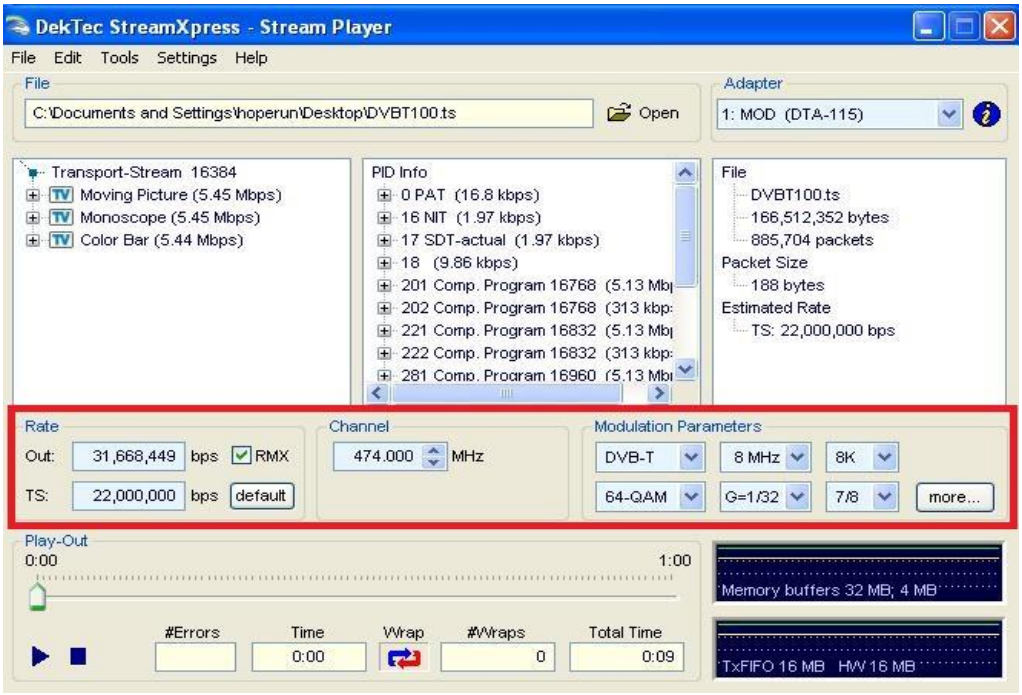
(2). Install“StreamXpress player”.

LINK: https://www.dektec.com/products/applications/StreamXpress/

(3). Open StreamXpress player and choose TS file.



(4). Setting Stream Player：

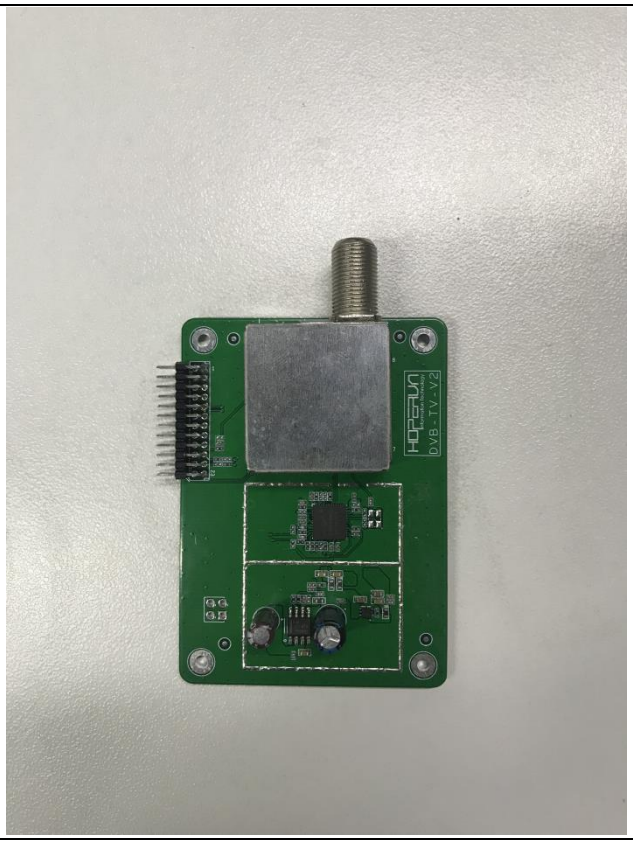


Ps:The value showed in this picture is the default config in tuner's driver.

(5). Play TS file:

4.2 Akebi96 hardware link

There ares three devices need here， there are akebi96 board, tuner board, antenna.

Akebi96 Board	
Tuner board	

Antenna	

Connect like this:

4.3 Install DVB Demos On akebi96 devices

1. download source codes

```
git clone https://github.com/hihope/akebi96-dvbdemos.git
```

2. Build

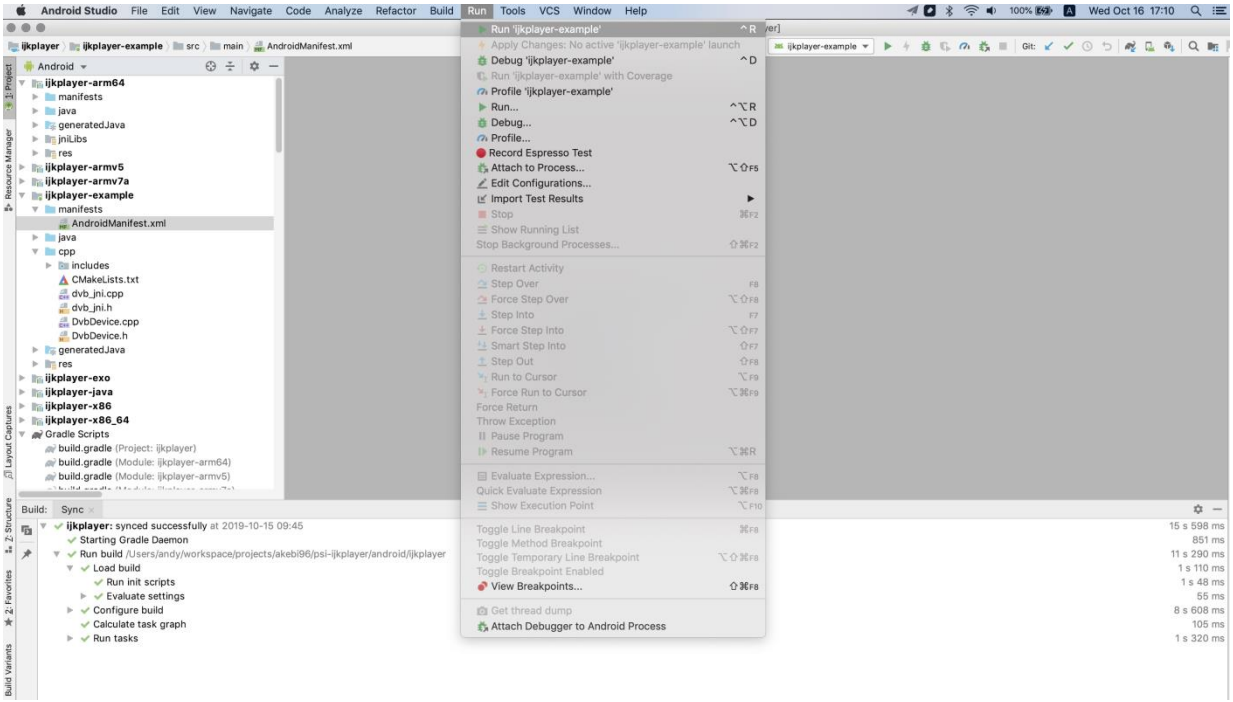
```
./init-android.sh

cd android/contrib
./compile-ffmpeg.sh clean
./compile-ffmpeg.sh all

cd ..
```

```
./compile-ijk.sh all

# Android Studio:
#         Open an existing Android Studio project
#         Select android/ijkplayer/ and import
#         Select Run/Run ijkplayer-examples
```



4.4 Run DVB Demos

Run demos on devices Following steps below :

Select Settings

Select Apps

Select Brow system apps

Select Show all apps

Select ijkplaydemos

Select Open

App will run like this :

4.5 Test Videos

demo videos url:
https://pan.baidu.com/s/1Pt_vJDRFkcpHj71uNldvow
password: 24an