Contents

[I. Host Development System: 2](#_Toc522633713)

[1.1. Ubuntu 14.04-64bit 2](#_Toc522633714)

[II. Build Linux Kernel for azalea\_r1576: 2](#_Toc522633715)

[III. Build Bootloader for azalea\_r1760 4](#_Toc522633716)

[3.1 Build Bootloader: 4](#_Toc522633717)

[3.2 Build early camera: 4](#_Toc522633718)

[IV. dolphin\_plus\_bsp 4](#_Toc522633719)

[4.1. Build U-boot: 4](#_Toc522633720)

[4.2. Build Linux 4.4.120: 5](#_Toc522633721)

[V. Driver porting 7](#_Toc522633722)

[5.1 Module driver porting outside Linux kernel source code. 7](#_Toc522633723)

[5.2 Driver is built-in Linux kernel. 8](#_Toc522633724)

# Host Development System:

## Ubuntu 14.04-64bit

* Install additional libraries:

$ sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0

* Install ARM toolchain: arm-2013.11 **for AArch32**
  + Download:

$ wget https://sourceforge.net/projects/epwa/files/arm-2013.11-33-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2

* + **Install toolchain into /opt/ for AArch32**

$ tar -xvf /home/thinhnt7/Downloads/arm-2013.11-33-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2

* Install ARM toolchain: **for AArch64**
  + **Download and install inside /opt/**

$ wget http://releases.linaro.org/12.10/components/toolchain/gcc-linaro/aarch64/rc3/gcc-linaro-aarch64-linux-gnu-4.7+bzr115029-20121015+bzr2506\_linux.tar.bz2

# Build Linux Kernel for azalea\_r1576:

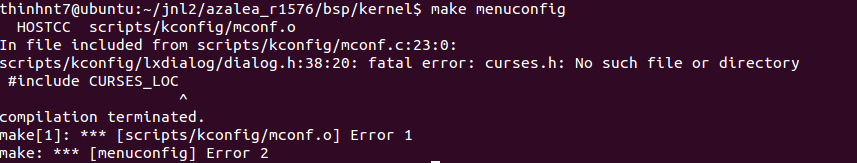
$ cd kernel/

$ export PATH=/opt/arm-2013.11/bin/:$PATH

$ export ARCH=arm CROSS\_COMPILE=/opt/arm-2013.11/bin/arm-none-linux-gnueabi-

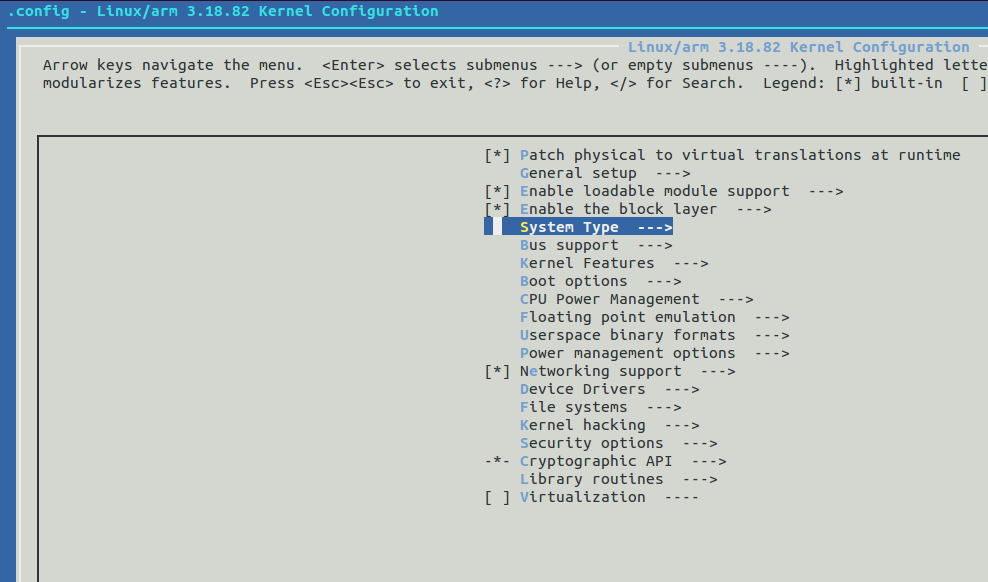
$ make jk\_19common\_fsbios\_defconfig

$ make menuconfig



**->Fix** $ sudo apt-get install libncurses5-dev libncursesw5-dev

$ make menuconfig



The custom configuration will be changed to save into the .config file for compiling.

Create a Linux kernel image:

$ make zImage

# Build Bootloader for azalea\_r1760

## Build Bootloader:

$ export PATH=/opt/arm-2013.11/bin/:$PATH #-> setup evn

$ cd azalea\_r1760/bsp/bootloader

$ make jk\_19m\_2048\_fsbios\_qboot

## Build early camera:

- Download toolchain to compile ARM cortex-M:

$ wget https://sourceforge.net/projects/epwa/files/arm-2013.11-24-arm-none-eabi-i686-pc-linux-gnu.tar.bz2

- Uncompressing them inside the directory "toolchain/gcc"

$ cd ./azalea\_r1760/bsp/bootloader/dev/camera/earlycamera

$ mkdir -p build/toolchain/gcc

$ tar -xvzf arm-2013.11-24-arm-none-eabi-i686-pc-linux-gnu.tar.bz2 buil/toolchain/gcc

- Build: (Notes: There is not bin2hex file for generating early camera file integrated inside bootloader)

$ make clean

$ make

# dolphin\_plus\_bsp

## 4.1. Build U-boot:

- Create a .config to build: notes Default is ARCH64

$ make ARCH=arm tcc803x\_defconfig

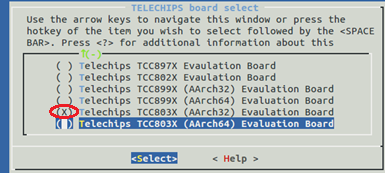
- If you want to change ARCH, follow up

$ make ARCH=arm menuconfig

And choose

ARM architecture --->

[\*] TELECHIPS board select (Telechips TCC803X (AArch32) Evaulation Board) --->



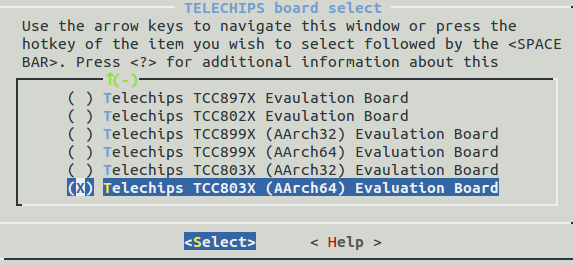
* + - Build for **AArch32**

$ make ARCH=arm CROSS\_COMPILE=arm-none-linux-gnueabi-

Notes: toolchain for AArch32 is arm-none-linux-gnueabi-

* + - Build for **AArch64**:

$ make ARCH=arm menuconfig ->select Arm architecture (AArch64)



**Notes: toolchain** for AArch64 is aarch64-linux-gnu-

$ make ARCH=arm CROSS\_COMPILE= aarch64-linux-gnu-

## 4.2. Build Linux 4.4.120:

* + - Toolchain: aarch64-linux-gnu-

Download toolchain: gcc-linaro-7.3.1-2018.05-i686\_aarch64-linux-gnu.tar.xz

Uncompressing/installing it inside /opt/

* + - Export PATH:

$ export PATH=/opt/gcc-linaro-7.3.1-2018.05-i686\_aarch64-linux-gnu/bin/:$PATH

* + - Create .config file for arm64

$ make ARCH=arm64 tcc803x\_linux\_avn\_defconfig

* + - Change any configuration if you need

$ make ARCH=arm64 menuconfig

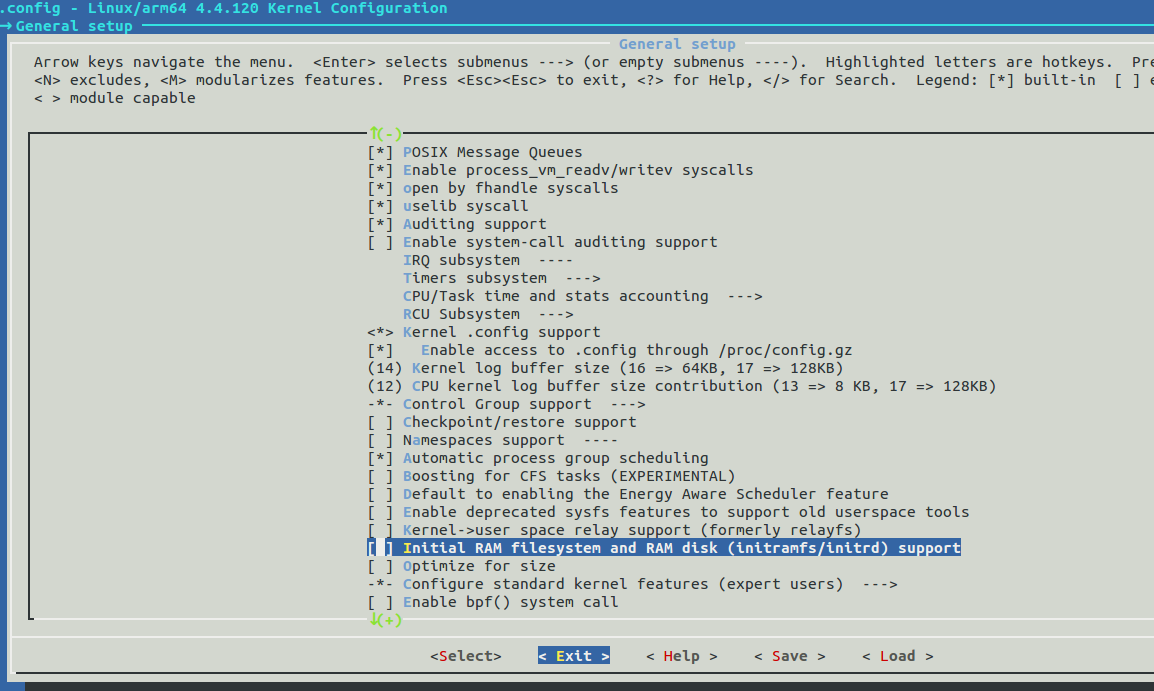
-Build kernel:

$ make ARCH=arm64 CROSS\_COMPILE=aarch64-linux-gnu-

* Error here:



-> How to fix: Enter “space” key to unselect initial RAM



* + - Build again:

Notes: bsp/kernel\_4.4/include/video/tcc/TCC\_HEVCDEC.h must keep as default of original source code.

Build:

$ make ARCH=arm64 CROSS\_COMPILE=aarch64-linux-gnu-

Reference: <https://wiki.linaro.org/HowTo/BuildArm64Kernel>

# Driver porting

New BSP

(dolphin\_plus\_bsp)

Copy source code

Fix build, Makefile, Kconfig, Hw config

Old BSP

(azalea\_r1760)

Note: you need to do the step at 4.2 before porting implement.

## Module driver porting outside Linux kernel source code.

Ex: porting driver “jk\_dummykey”

|  |  |
| --- | --- |
| Old source location | azalea\_r1760/bsp/kernel/drivers/input/keyboard/jk\_dummykey.c |
| New source location | Everywhere in your host pc.  Ex: /home/thinhnt7/thinhnt7/custom\_module/jk\_dummykey/ jk\_dummykey.c Makefile |

jk\_dummykey.c: is a device driver module copied from old source to NEW location.

Change Makefile:

|  |
| --- |
| # Run on ARM Platform  export ARCH=arm64  export CROSS\_COMPILE=aarch64-linux-gnu-  KDIR ?=/home/thinhnt7/thinhnt7/client\_source\_code/dolphin\_plus\_bsp/bsp/kernel\_4.4  # Run on PC  #KDIR ?=/lib/modules/$(shell uname -r)/build  # only one source file  obj-m += jk\_dummykey.o  # module has several source file  #obj-m += MscDev.o  #MscDev-objs += MscDev.o MainComm.o PanelComm.o  #build:  all:  make -C $(KDIR) M=$(PWD) modules  clean:  make -C $(KDIR) M=$(PWD) clean |

Switch to source location and set PATH for toolchain:

$ cd /home/thinhnt7/thinhnt7/custom\_module/jk\_dummykey

$ export PATH=/opt/gcc-linaro-7.3.1-2018.05-i686\_aarch64-linux-gnu/bin/:$PATH

Build module and clean:

$ make clean

$ make

Compile result: 

## Driver is built-in Linux kernel.

Ex: porting driver “jk\_dummykey”

Step 1: Coping source code from old BSP to new BSP

azalea\_r1760/bsp/kernel/drivers/input/keyboard/jk\_dummykey.c

* dolphin\_plus\_bsp/bsp/kernel\_4.4/drivers/input/keyboard/

**Step 2:** Add build condition in Makefile:

Opening the Makefile /azalea\_r1760/bsp/kernel/drivers/input/keyboard/Makefile

Coping obj-$(CONFIG\_KEYBOARD\_JK\_DUMMYKEY) += jk\_dummykey.o

* Pasting into the Makefile:

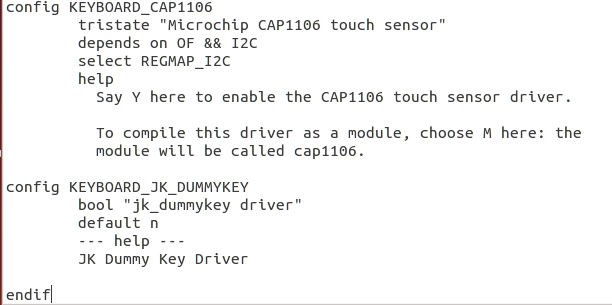
dolphin\_plus\_bsp/bsp/kernel\_4.4/drivers/input/keyboard/ Makefile



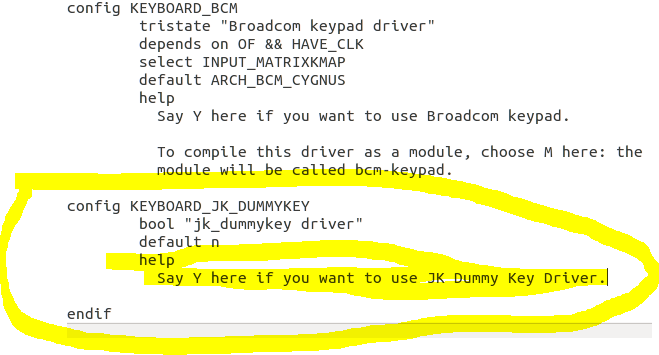
**Step 3:** Update Kconfig file for creating a menu configuration:

Opening Kconfig in old source location and copy contents of “**KEYBOARD\_JK\_DUMMYKEY**” into the Kconfig file of new BSP as following samle.

Old munuconfig on Kernel 3.18.82



New menuconfig on Kernel 4.4.120:



**Step 4: Check menuconfig on new BSP:**

$ make ARCH=arm64 menuconfig

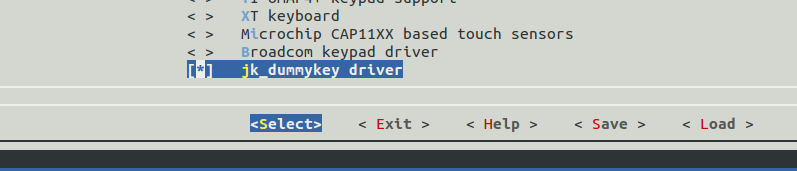
Selecting as path:

**Device Drivers --->**

**Input device support --->**

**[\*] Keyboards --->**

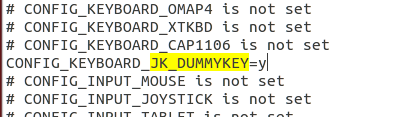
And slide down cursor to **select** jk\_dummykey\_driver and save this configuration



**Step 5:** Check configuration have selected into .config file

**$ gedit .config**

see CONFIG\_KEYBOARD\_JK\_DUMMYKEY=y in .config file



* **© Porting is successful.**

**Step 6:** Store all configurations into /arch/arm\*/xxx\_defconfig

$ make ARCH=arm64 savedefconfig

* Generate a defconfig file in root directory of kernel “/’

$ cp defconfig /arch/ arm64 /configs/**custom\_board**\_defconfig