ANDROID12 GENERIC KERNEL IMAGE (GKI) DEVELOPMENT TIPS

CAS MAGGIE JIANG

2022 DEC

SECURE CONNECTIONS FOR A SMARTER WORLD

Topics

From Android 12, NXP released i.MX BSP is based on Generic Kernel Image(GKI). It means that it is not NXP kernel image, instead it is Android standard GKI. Customer need to be careful during development.

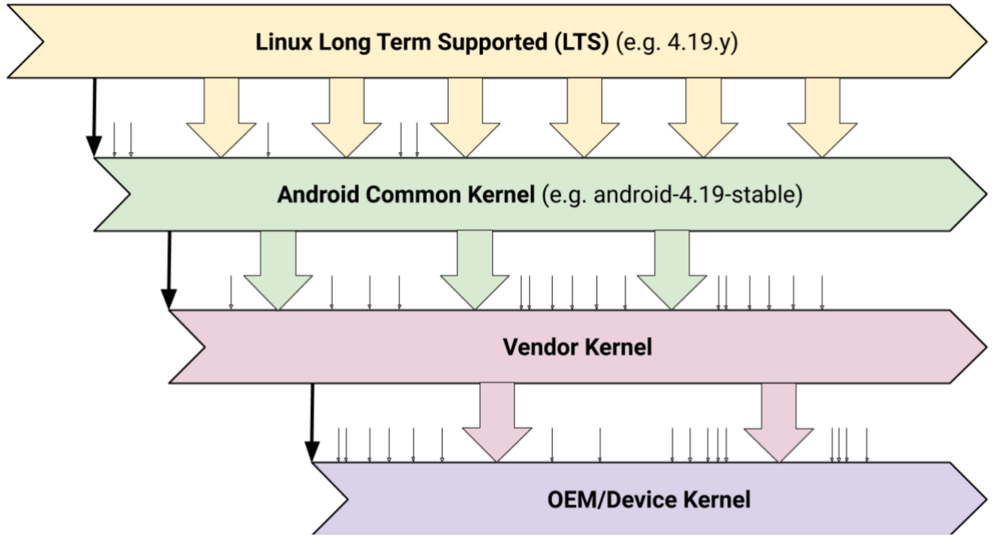
This document describe how customer develop their own kernel based on GKI.

Android GKI structure document:

https://source.android.com/docs/core/architecture/kernel/generic-kernel-image

Previous Android kernel code fragmentation issue

Android kernel hierarchy leads to fragmentation(碎片化问题）



Linux Long Term Supported (LTS) (e.g. 4.19.y)
Android Common Kernel (e.g. android-4.19-stable)
Vendor Kernel
OEM/Device Kernel

Security updates are difficult.

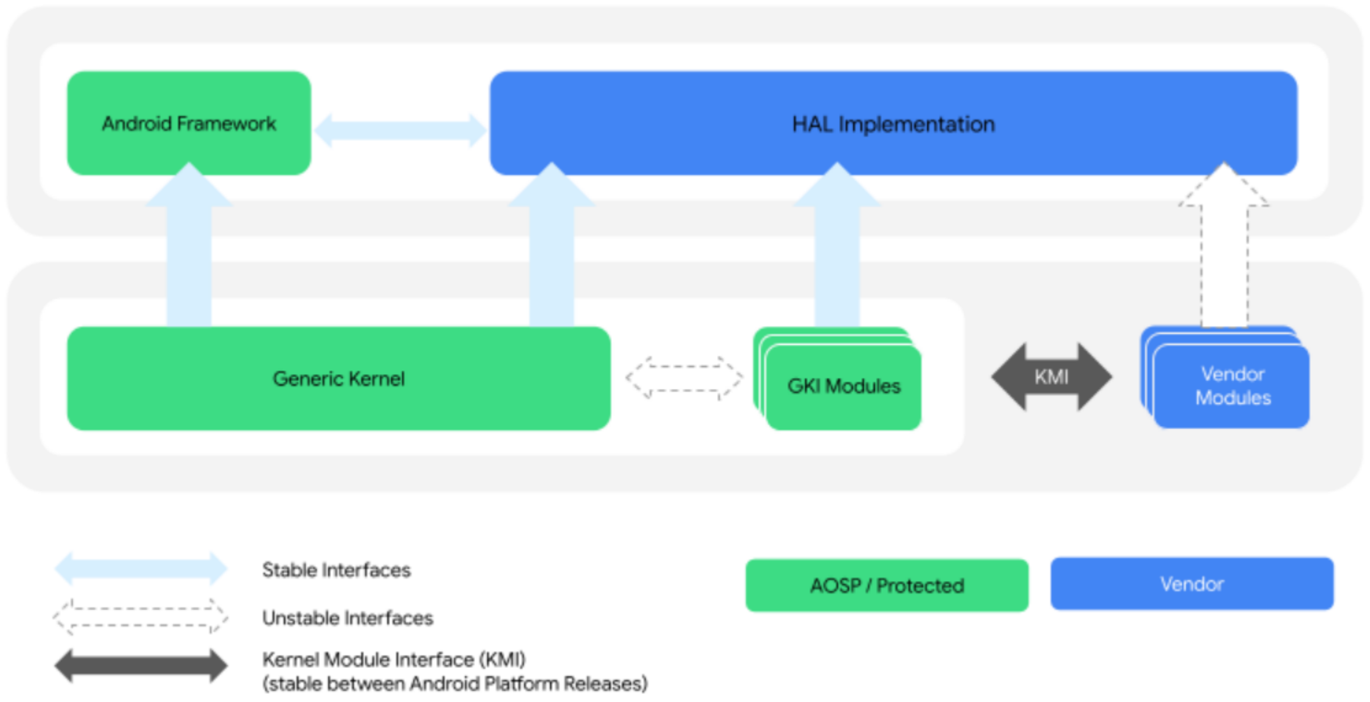
Difficult to merge Long-Term Supported update

Difficult to contribute kernel changes back to upstream Linux

………

Fixing the fragmentation: Generic Kernel Image

- The Generic Kernel Image (GKI) project addresses kernel fragmentation by unifying the core kernel and moving SoC and board support out of the core kernel into loadable modules. The GKI kernel presents a stable Kernel Module Interface (KMI) for kernel modules, so modules and kernel can be updated independently.



Android 12 environment setup

Download Android 12 code:

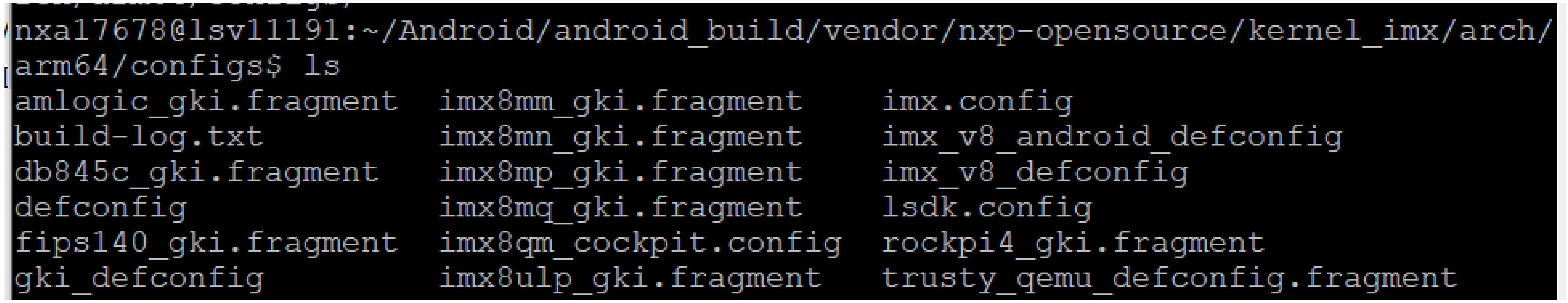
```
$ mkdir ~/bin
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=${PATH}:~/bin
$ source ~/imx-android-12.0.0_2.0.0/imx_android_setup.sh
```

Build i.MX Android image:

```
cd ${MY_ANDROID}
$ source build/envsetup.s
$ lunch evk_8mm-userdebug
$ ./imx-make.sh -j4 2>&1 | tee build-log.txt
```

Android 12 Kernel config file:

- Kernel config file directory is under /android_build/vendor/nxp-opensource/kernel_imx/arch/arm64/configs



```
nxa17678@lsv11191:~/Android/android_build/vendor/nxp-opensource/kernel_imx/arch/
arm64/configs$ ls
amlogic_gki.fragment   imx8mm_gki.fragment    imx.config
build-log.txt          imx8mn_gki.fragment    imx_v8_android_defconfig
db845c_gki.fragment    imx8mp_gki.fragment    imx_v8_defconfig
defconfig              imx8mq_gki.fragment    lsdk.config
fips140_gki.fragment   imx8qm_cockpit.config  rockpi4_gki.fragment
gki_defconfig          imx8ulp_gki.fragment   trusty_qemu_defconfig.fragment
```

- From Android 12, Kernel config file is composed of by gki_defconfig and imx8mm_gki.fragment two files together, since it use GKI.
- gki_defconfig file is standard kernel config.
- imx8mm_gki.fragment is i.MX8MM related config file. In this file, Almost vendor i.MX related modules are M:

CONFIG_I2C_IMX=m

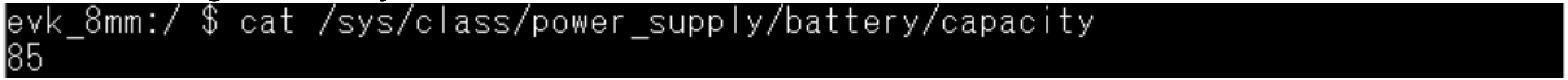
CONFIG_IMX_SDMA=m

CONFIG_IMX8MM_THERMAL=m

Debug case example

In android-12.0.0_2.0.0 demo image, the status is following:

1. We can get battery information 80.



```
evk_8mm:/ $ cat /sys/class/power_supply/battery/capacity
85
```

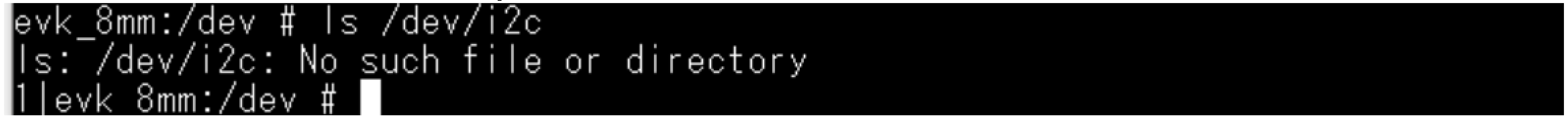
This value come from NXP kernel code:

android_build/vendor/nxp-opensource/kernel_imx/drivers/power/supply$ vi dummy_battery.c

#define POWER_SUPPLY_PROP_CAPACITY_VALUE 85

This is a dummy battery driver since there is no real battery on evk.

2. There is no I2C access point under /dev.



```
evk_8mm:/dev # ls /dev/i2c
ls: /dev/i2c: No such file or directory
1|evk_8mm:/dev #
```

Modify kernel driver and config file

Now we want to change this dummy battery information from 85 to 60, and add i2c assess point to /dev/.

Then we modify following two points:

1. android_build/vendor/nxp-opensource/kernel_imx/drivers/power/supply$ vi dummy_battery.c

#define POWER_SUPPLY_PROP_CAPACITY_VALUE 60

2. Modify i.MX8MM kernel config file imx8mm_gki.fragment.

/android_build/vendor/nxp-opensource/kernel_imx/arch/arm64/configs/imx8mm_gki.fragment

 CONFIG_I2C_CHARDEV=Y

Question： Some kernel modification didn’t make effect

- build the whole android image

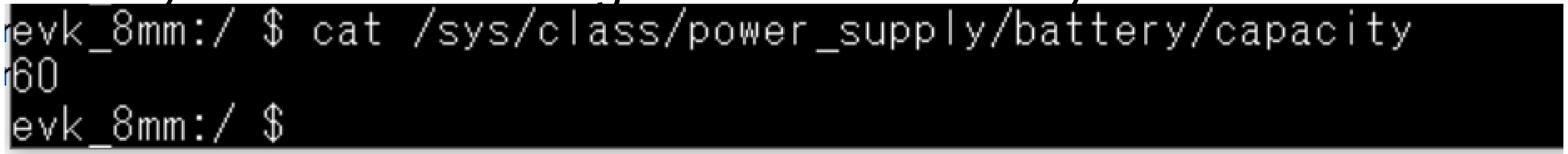
$ ./imx-make.sh -j4 2>&1 | tee build-log.txt

- download the whole image:

uuu_imx_android_flash.bat -f imx8mm -a –e

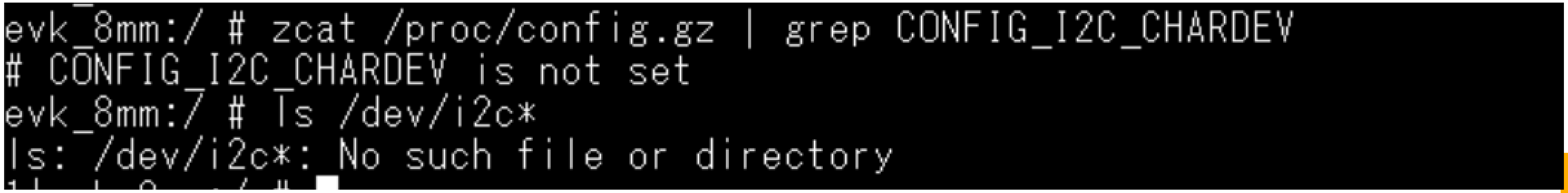
- Check board status:

1. Battery information change to 60 successfully



```
evk_8mm:/ $ cat /sys/class/power_supply/battery/capacity
60
evk_8mm:/ $
```

2. Config file didn’t make effect and there is still no I2C access point



```
evk_8mm:/ # zcat /proc/config.gz | grep CONFIG_I2C_CHARDEV
# CONFIG_I2C_CHARDEV is not set
evk_8mm:/ # ls /dev/i2c*
ls: /dev/i2c*: No such file or directory
```

Two kernel image difference

In Android demo image and build directory android_build/out/target/product/evk_8mm, there is boot.img and boot-imx.img， which are different.

- boot.img: a composite image which includes the AOSP generic kernel Image, generic ramdisk and boot parameters. It is GKI kernel image actually.
- boot-imx.img: a composite image which includes the kernel Image built from i.MX Kernel tree, generic ramdisk and boot parameters. It is NXP kernel image actually.

when use uuu tool to download, the log is following, it download GKI boot.img into board, instead of NXP boot image.

- uuu_imx_android_flash.bat -f imx8mm -a -e

log output:

```
……………………………………………………..
generate lines to flash boot.img to the partition of boot_a
…………………………………………………….
```

Android makefile to produce boot.img and boot-imx.img

Build android command line: ./imx-make.sh -j4 2>&1 | tee build-log.txt

Let’s check imx-make.sh file, it produce both boot.img and boot-imx.img. uuu download use boot.img

```
TARGET_IMX_KERNEL=true make ${parallel_option} ${build_bootimage} ${build_vendorbootimage} ${build_dtboimage} ${build_vendorimage} ||
exit
```

# With TARGET_IMX_KERNEL=true, this step produce boot.img which is NXP kernel image.

```
  if [ -n "${build_bootimage}" ] || [ ${build_whole_android_flag} -eq 1 ]; then
      if [ ${TARGET_PRODUCT} = "evk_8mp" ] || [ ${TARGET_PRODUCT} = "evk_8mn" ] \
      || [ ${TARGET_PRODUCT} = "evk_8ulp" ] \
      || [ ${TARGET_PRODUCT} = "evk_8mm" ] || [ ${TARGET_PRODUCT} = "evk_8mq" ]; then
          mv ${OUT}/boot.img ${OUT}/boot-imx.img
```

# If it is i.MX EVK board, then change name from boot.img to boot-imx.img. So boot-imx.img is NXP kernel image now.

```
          # sign prebuilt gki boot.img
          make bootimage
```

- **# rebuild bootimage again. So boot.img is GKI kernel image now.**

Android 12 image status

1.Boot.img is default GKI image, almost all build in kernel are inside this image. GKI source code are not included in Anroid12 build environment.

2.i.MX related kernel are all built in with module type. Module sour code are located in android_build/vendor/nxp-opensource/kernel_imx.

3.If customer want to add new driver, it can only be built into module. They can't build into kernel.

4. When compile Android and use uuu to download to board, it use boot.img which is GKI image.

5. Dts modification is still in NXP kernel code, because dtb file is separately from boot.img.

/android_build/vendor/nxp-opensource/kernel_imx/arch/arm64/boot/dts/freescale

•

Debug case failed reason

- Let’s check previous debug case:

Why modifying android_build/kernel_imx/drivers/power/supply/dummy_battery.c take effect, while

modifying kernel_imx/arch/arm64/configs/imx8mm_gki.fragment have no effect?

The reason is：

1. Dummy battery driver are built into module by default(imx8mm_gki.fragment :CONFIG_BATTERY_DUMMY=m),so we can modify NXP module code and it take effect. If this driver is built into kernel, then it is no effect to modify NXP kernel code.
2. It is no useful to add a new config CONFIG_I2C_CHARDEV=Y in config file, because it use Android standard GKI kernel image. Boot.img is default GKI image.

How to modify kernel config file (way1:use NXP kernel, just for test purpose)

There are two ways to add CONFIG_I2C_CHARDEV to config file:

1.Use NXP kernel code and build into NXP kernel:

① /android_build/vendor/nxp-opensource/kernel_imx/arch/arm64/configs/ imx8mm_gki.fragment : CONFIG_I2C_CHARDEV=Y

② Build android all images: ./imx-make.sh -j4 2>&1 | tee build-log.txt

③ Add this procedure to rebuild NXP kernel image again, then boot.img is NXP kernel image to replace default GKI image: TARGET_IMX_KERNEL=true make bootimage

2.Test on board, it prove that config file take effect and there is I2C access point.

```
evk_8mm:/ $ zcat /proc/config.gz | grep CONFIG_I2C_CHARDEV

  CONFIG_I2C_CHARDEV=y

evk_8mm:/dev $ ls /dev/i2c*
/dev/i2c-0  /dev/i2c-1  /dev/i2c-2
```

How to modify kernel config file(way2: use GKI kernel)

The official way is use GKI kernel and build new driver into module.

1.android_build/vendor/nxp-opensource/kernel_imx/drivers/i2c$ vi Makefile

```
bj-$(CONFIG_I2C_CHARDEV)        += i2c-dev.o
```

2. android_build/vendor/nxp-opensource/kernel_imx/arch/arm64/configs/ imx8mm_gki.fragment: CONFIG_I2C_CHARDEV=M

3. android_build/device/nxp/imx8m/evk_8mm$ vi SharedBoardConfig.mk

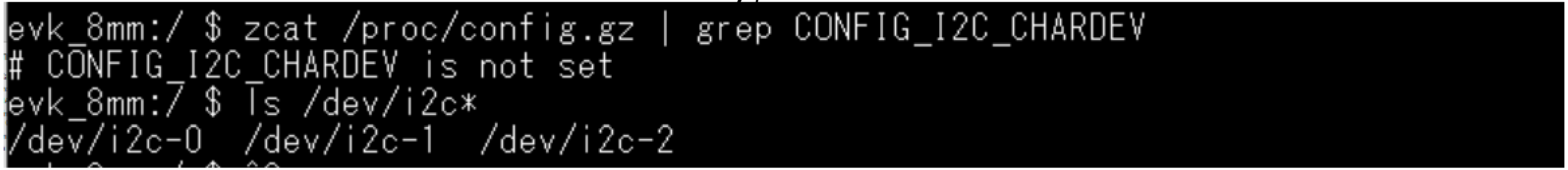
```
$(KERNEL_OUT)/drivers/i2c/i2c-dev.ko
```

SharedBoardConfig.mk list all modules loaded automatically when bootup, so customer need to add a new module here, to let system to load it when bootup.

How to modify kernel config(way2:use GKI kernel, board check)

Build android all images: ./imx-make.sh -j4 2>&1 | tee build-log.txt

Then i2c-dev.ko is loaded automatically,



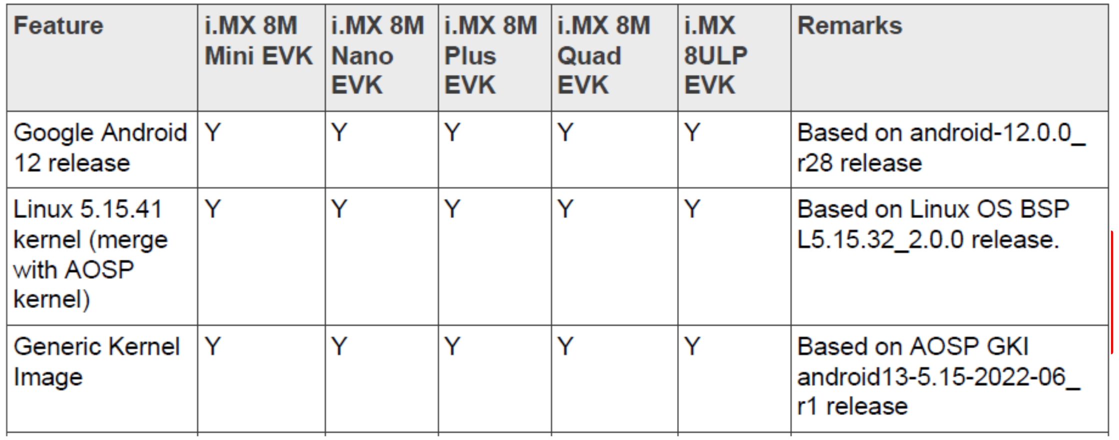
```
evk_8mm:/ $ zcat /proc/config.gz | grep CONFIG_I2C_CHARDEV
# CONFIG_I2C_CHARDEV is not set
evk_8mm:/ $ ls /dev/i2c*
/dev/i2c-0  /dev/i2c-1  /dev/i2c-2
```

- Although CONFIG_I2C_CHARDEV is not set in config, it is set in module and loaded automatically, we can access I2C device now.

How to download GKI source code

Sometimes customer need to modify GKI code. This normally only happened when need to modify Android standard kernel. Normally, customer won't modify it.

From Android 12 release notes, it said GKI kernel is android13-5.15-2022-06_r1 release:

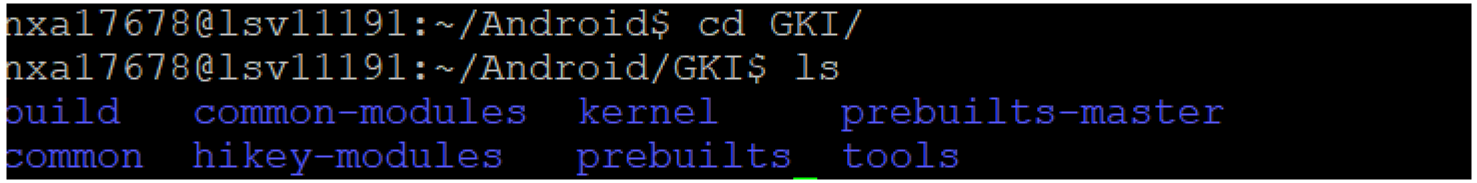


Feature	i.MX 8M Mini EVK	i.MX 8M Nano EVK	i.MX 8M Plus EVK	i.MX 8M Quad EVK	i.MX 8ULP EVK	Remarks
Google Android 12 release	Y	Y	Y	Y	Y	Based on android-12.0.0_ r28 release
Linux 5.15.41 kernel (merge with AOSP kernel)	Y	Y	Y	Y	Y	Based on Linux OS BSP L5.15.32_2.0.0 release.
Generic Kernel Image	Y	Y	Y	Y	Y	Based on AOSP GKI android13-5.15-2022-06_ r1 release

Then we go to https://android.googlesource.com/kernel/manifest and check the branch, it should be common-android13-5.15-2022-06

How to download GKI source code

- mkdir gki && cd gki
- repo init -u https://android.googlesource.com/kernel/manifest -b common-android13-5.15-2022-06
- repo sync
- GKI Kernel tree is located in : gki/common， go to this directory, to get android13-5.15-2022-06_r1 tag.



```
nxa17678@lsv11191:~/Android$ cd GKI/
nxa17678@lsv11191:~/Android/GKI$ ls
build    common-modules  kernel      prebuilts-master
common   hikey-modules   prebuilts   tools
```

nxa17678@lsv11191:~/Android/GKI/common$ git checkout -b android13-5.15-2022-06_r1
Switched to a new branch 'android13-5.15-2022-06_r1'

(In Android 12 user guide, it give an example on GKI download address, which is not aligned with Android 12. So following address is wrong)

(repo init -u https://android.googlesource.com/kernel/manifest -b common-android12-5.10)

Compile GKI source code

Build the GKI Kernel Image:

- BUILD_CONFIG=common/build.config.gki.aarch64 build/build.sh

Copy GKI Image to Android build directory.

- cp gki/out/android12-5.10/common/arch/arm64/boot/Image

${MY_ANDROID}/out/target/product/evk_8mm/obj/KERNEL_OBJ/arch/arm64/boot/Image

Rebuild Android image:

- cd ${MY_ANDROID}
- TARGET_IMX_KERNEL=true make bootimage

Conclusions on how to develop kernel based on GKI

1. If customer apply standard kernel patch which is not i.MX related, such as ARM related patch, security vulnerability patch, some kernel upstream patch, they need to download standard GKI source code firstly, then add patch, to produce GKI kernel image firstly.

2. If customer need to modify i.MX driver related code, they need to make sure it is built in module, then modify NXP kernel code under Android build environment.

3. If customer add a new device into kernel config, then they must build it as module.

4. We don’t suggest customer to use NXP kernel code. The reason is:

① GKI is Android standard kernel structure, we should follow it;

② What NXP tested is based on GKI image, instead of NXP kernel code. There are some modifications from bsp into common code. And NXP tested system is based on boot.img instead of boot-imx.img. So maybe there will be risks, if customer use NXP kernel code.

SECURE CONNECTIONS
FOR A SMARTER WORLD