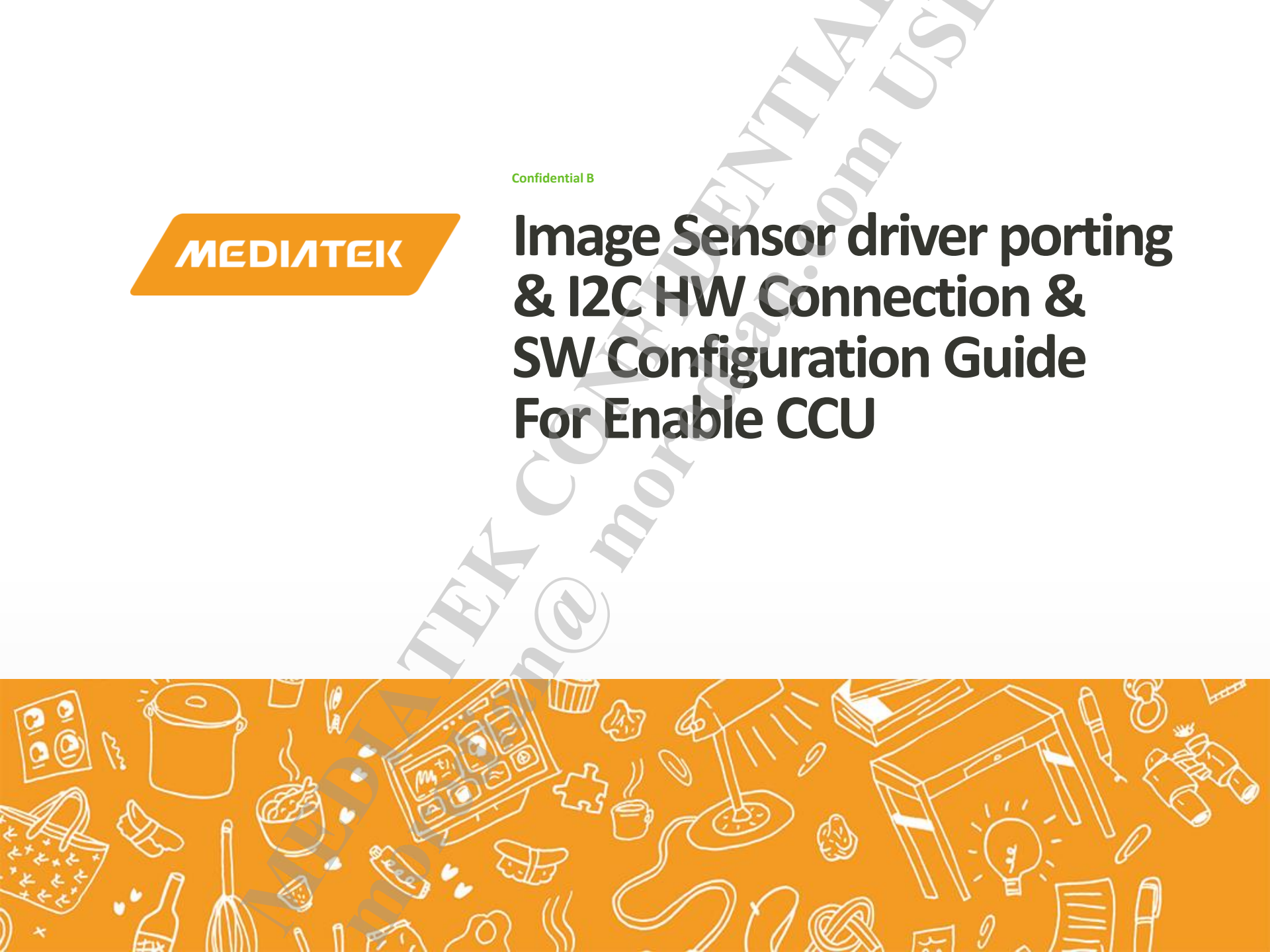


**Confidential B**

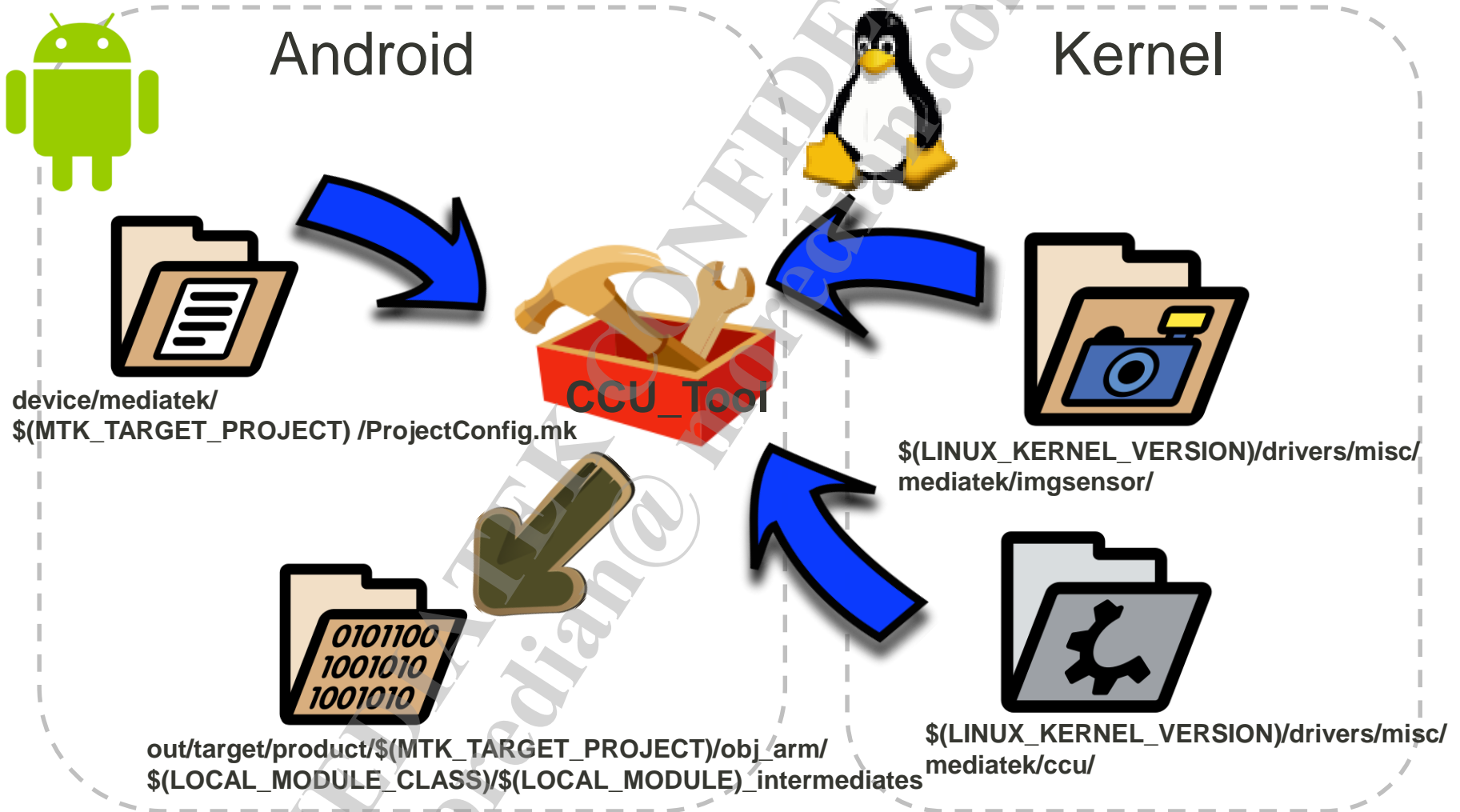
# Image Sensor driver porting & I2C HW Connection & SW Configuration Guide For Enable CCU



# Agenda

- **Overview for CCU build system**
- **Generation of CCU sensor driver**
  - Sensor driver porting guide
    - Rule 1: Naming rule while creating a new sensor driver
    - Rule 2: The file name of major image sensor header file must be the same as .c file
    - Rule 3: Do not use non CCU supported functions
    - Rule 4: Do not use undefined macro
  - Build CCU sensor driver
  - Trouble shooting
- **I2C HW architecture**
  - Mediatek I2C bus/controller architecture
  - CCU/APMCU sensor I2C controlling scenario
- **I2C-Bus Connection constraint of image sensor to activate CCU**
- **CCU I2C controller SW configuration**
  - How-to: configure I2C controller SW settings for CCU

# Overview



# Sensor driver porting guide

# Customization Guideline of image sensor for CCU

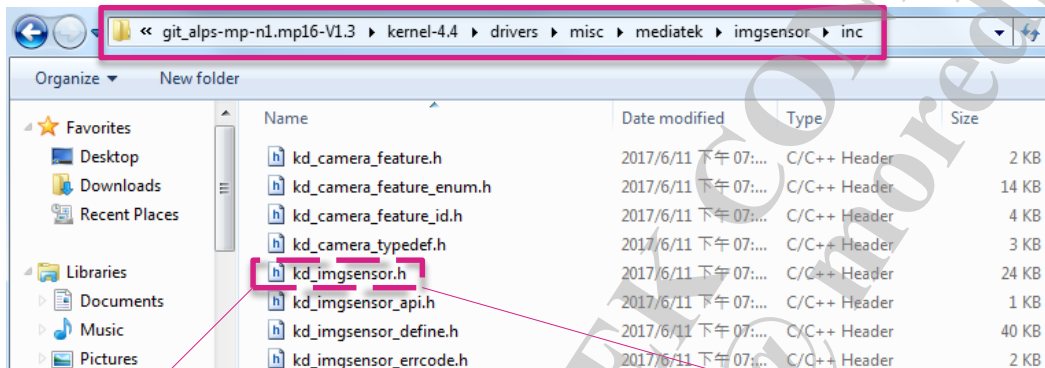
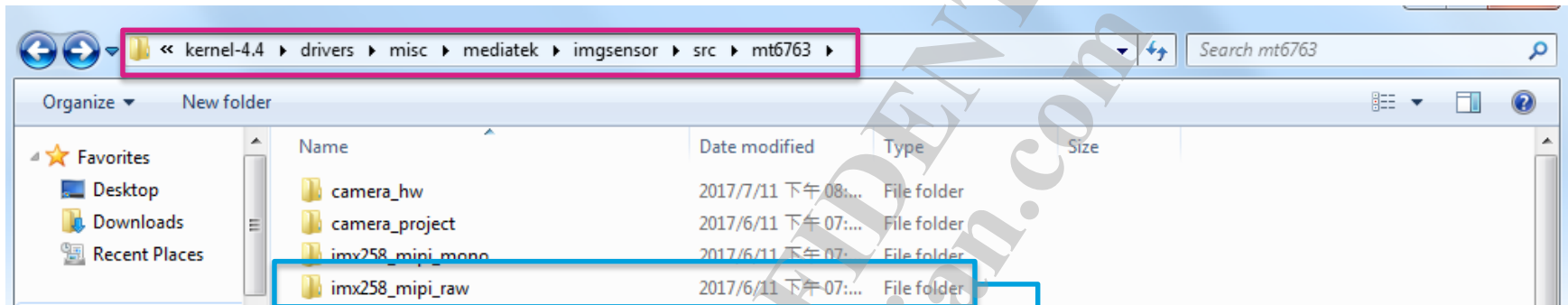
- **Rule 1: Create a new sensor driver**
  - No matter what name of filename extension you want to have, **the sensor driver name** has to be **the same** as defined at ***"kd\_imgsensor.h"***.

If violate the rule, it will compiled error with:

**"#error "Violate CCU sensor driver coding RULE 1: Cant find source code"**

**"#error "Violate CCU sensor driver coding RULE 1: Not defined in kd\_imgsensor.h"**

# Example of Create a new sensor driver



Equal to

#define SENSOR\_DRVNAME IMX258\_MIPI\_RAW

Format:  
SENSOR\_DRVNAME\_ + <sensor\_name>

Upper case

# Customization Guideline of image sensor for CCU

- Rule 2: The header file has to exist, and should be same to C file after removing suffix.

The right case:

imx258_pdafotp.c	2017/6/11 下午 07:...	C Source	4 KB
imx258mipiraw_Sensor.c	2017/6/11 下午 07:...	C Source	117 KB
imx258mipiraw_Sensor.h	2017/6/11 下午 07:...	C/C++ Header	6 KB
Makefile	2017/6/11 下午 07:...	File	1 KB

The wrong case:

imx258_pdafotp.c	2017/6/11 下午 07:...	C Source	4 KB
imx258mipiraw_Sensor_test.c	2017/6/11 下午 07:...	C Source	117 KB
imx258mipiraw_Sensor.h	2017/6/11 下午 07:...	C/C++ Header	6 KB
Makefile	2017/6/11 下午 07:...	File	1 KB

If violate the rule, it will compiled error with:

**"#error "Violate CCU sensor driver coding RULE 2: %s does not exist"**

**or see the build log:**

```
/bin/bash: line 1: 19818 Segmentation fault (core dumped)
ccu_tool/mt6771/ver1/extractor kernel-
4.4/drivers/misc/mediatek/imgsensor/src
/mt6771/camera_project/k71v1_64_bsp/s5k4h7yx_mipi_raw/s5k4h7
yxmipi_Sensor.c > /dev/null
```

```
s5k4h7yx_mipi_raw: is_ccu_supported =
```



# Customization Guideline of image sensor for CCU

## ■ Rule 3: Using non CCU supported function

- After processing by ccu\_tool, CCU sensor driver will be compiled by md32 compiler
- CCU sensor driver run in CCU, so can not link functions that are implemented out of this sensor driver file except the functions of white list
  - White list: Already implemented in CCU
    - iWriteRegI2C
    - iWriteRegI2CTiming
    - iReadRegI2C
    - iReadRegI2CTiming
    - fls

If violate the rule, it will compiled error with:

**"#error "Violate CCU sensor driver coding RULE 3: using non CCU supported function"**

# Customization Guideline of image sensor for CCU

## ■ Rule 4: Using undefined macro

- Macros that are used in sensor driver should be declared specifically in “.c” or “.h” file that meet RULE 1.
  - Each macro should be defined as “#define” or “#undef”
- If the macro is defined in another file such as makefile or other header file, it will lead to a compile error.

If you violate the rule, it will cause a compile error with:

**"#error "Violate CCU sensor driver coding RULE 4: using undefined macro"**

# Build CCU sensor driver

# Build of CCU sensor driver (mt6771/mt6775)

## Remove the old build file in out folder


- `$out/target/product/$platform/obj_arm/EXECUTABLES/libccu_$sensor_name.ldr_intermediates`
- `$out/target/product/$platform/obj_arm/EXECUTABLES/libccu_$sensor_name_intermediates`

## Partial build ccu\_tool


## Push CCU sensor binary to phone(system/vendor/bin)

- `$out/target/product/$platform/vendor/bin/libccu_$sensor_name.ldr`

## Or full build directly

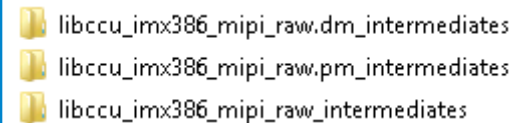


libccu\_imx386\_mipi\_raw.ldr\_intermediates  
libccu\_imx386\_mipi\_raw\_intermediates



libccu\_imx386\_mipi\_raw.ldr

# Build of CCU sensor driver (mt6758/mt6763/mt6765)



- libccu\_imx386\_mipi\_raw.dm\_intermediates
- libccu\_imx386\_mipi\_raw.pm\_intermediates
- libccu\_imx386\_mipi\_raw\_intermediates

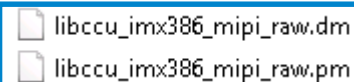
## Remove the old build file in out folder

- `$out/target/product/$platform/obj_arm/EXECUTABLES/libccu_$sensor_name.dm_intermediates`
- `$out/target/product/$platform/obj_arm/EXECUTABLES/libccu_$sensor_name.pm_intermediates`
- `$out/target/product/$platform/obj_arm/EXECUTABLES/libccu_$sensor_name_intermediates`

## Partial build ccu\_tool

## Push CCU sensor binary to phone(system/vendor/bin)

- `$out/target/product/$platform/vendor/bin/libccu_$sensor_name.dm`
- `$out/target/product/$platform/vendor/bin/libccu_$sensor_name.pm`



- libccu\_imx386\_mipi\_raw.dm
- libccu\_imx386\_mipi\_raw.pm

## Or full build directly

# Troubleshooting 1/2

- How to know a sensor is working in CCU ?
  - adb command : **adb shell 'logcat | grep -i ccu'**

```
CcuDrv[loadSensorBin] +=loadSensorBin
CcuDrv[GetSensorName] ccu main sensor name: imx398_mipi_raw
CcuDrv[GetSensorName] ccu sub sensor name: s5k4e6_mipi_raw
CcuDrv[GetSensorName] ccu main2 sensor name: imx350_mipi_raw
CcuDrv[loadSensorBin] ccu sensor name: imx398_mipi_raw
CcuDrv[loadSensorBin] Load Sensor DM
CcuDrv[loadBin] +=loadBin
CcuDrv[loadBin] open Bin file, path: /system/vendor/bin/libccu_imx398_mipi_raw.ldr
CcuDrv[tryOpenFile] open file, path: /system/vendor/bin/libccu_imx398_mipi_raw.ldr
CcuDrv[loadBin] open Bin file result:-228573172
CcuDrv[loadBin] read Bin file into Bin buffer
CcuDrv[loadSensorDrvToBuffer] SLOT = 1
CcuDrv[loadSensorDrvToBuffer] read file into buffer, szTell=8000
CcuDrv[loadSensorDrvToBuffer] Before seek=0
CcuDrv[loadSensorDrvToBuffer] After seek=0
CcuDrv[loadSensorDrvToBuffer] read file done
CcuDrv[loadBin] read Bin done
CcuDrv[loadBin] clear MEM & load Bin buffer onto MEM
CcuDrv[clearAndLoadBinToMem] clear MEM
CcuDrv[clearAndLoadBinToMem] args: 0xf142d000, 0xdalc68bc, 0, 4000, 38000
CcuDrv[clearAndLoadBinToMem] load bin buffer onto MEM
CcuDrv[clearAndLoadBinToMem] f801800,8f601003,78e07ee0
CcuDrv[loadBin] -=loadBin
CcuDrv[loadSensorBin] -=loadSensorBin
```

1 Sensor module used currently

2 CCU sensor driver binary exist

3 CCU sensor driver binary not exist

```
CcuDrv[GetSensorName] Matching Filters: r name: imx398_mipi_raw
CcuDrv[GetSensorName] Unnamed filter (c): name: s5k4e6_mipi_raw
CcuDrv[GetSensorName] ccu main sensor name: imx350_mipi_raw
CcuDrv[loadSensorBin] ccu sensor name: imx398_mipi_raw
CcuDrv[vendor/mediatek/proprietary/hardware/libcamera_ext/ccu/mt6771/ver1/drv/src/drv/ccu_udrv.cpp, tryOpenFile, line126] ERROR: open file fail: 0
CcuDrv[vendor/mediatek/proprietary/hardware/libcamera_ext/ccu/mt6771/ver1/drv/src/drv/ccu_udrv.cpp, tryOpenFile, line127] ERROR: open file path: /system/vendor/bin/libccu_imx398_mipi_raw.ldr
CcuDrv[vendor/mediatek/proprietary/hardware/libcamera_ext/ccu/mt6771/ver1/drv/src/drv/ccu_udrv.cpp, loadBin, line1266] ERROR: Open Bin file fail
```

3

# Troubleshooting 2/2

- How to check if sensor driver binary is correct or not
  - Search string “Table size mismatch!” , if it is found, it means contents of sensor binary is corrupted.
  - Search string “\_sensor\_init\_done”, if it is found, it means the sensor initial flow is done.

# I2C-Bus Connection constraint of image sensor to activate CCU



# I<sup>2</sup>C HW connection table

Platform	I <sup>2</sup> C architecture	I <sup>2</sup> C bus of APMCU	I <sup>2</sup> C controller/channel ID of CCU
MT6763	Multi-channel	I2C_BUS_2	I2C_CHANNEL_8
		I2C_BUS_4	I2C_CHANNEL_9
MT6765/MT6762	Multi-channel	I2C_BUS_2	I2C_CHANNEL_2
		I2C_BUS_4	I2C_CHANNEL_4
MT6771	Multi-channel	I2C_BUS_2	I2C_CHANNEL_2
		I2C_BUS_4	I2C_CHANNEL_4
MT6775	Multi-channel	I2C_BUS_2	I2C_CHANNEL_2
		I2C_BUS_4	I2C_CHANNEL_4

例如MT6763，根据上面表格：

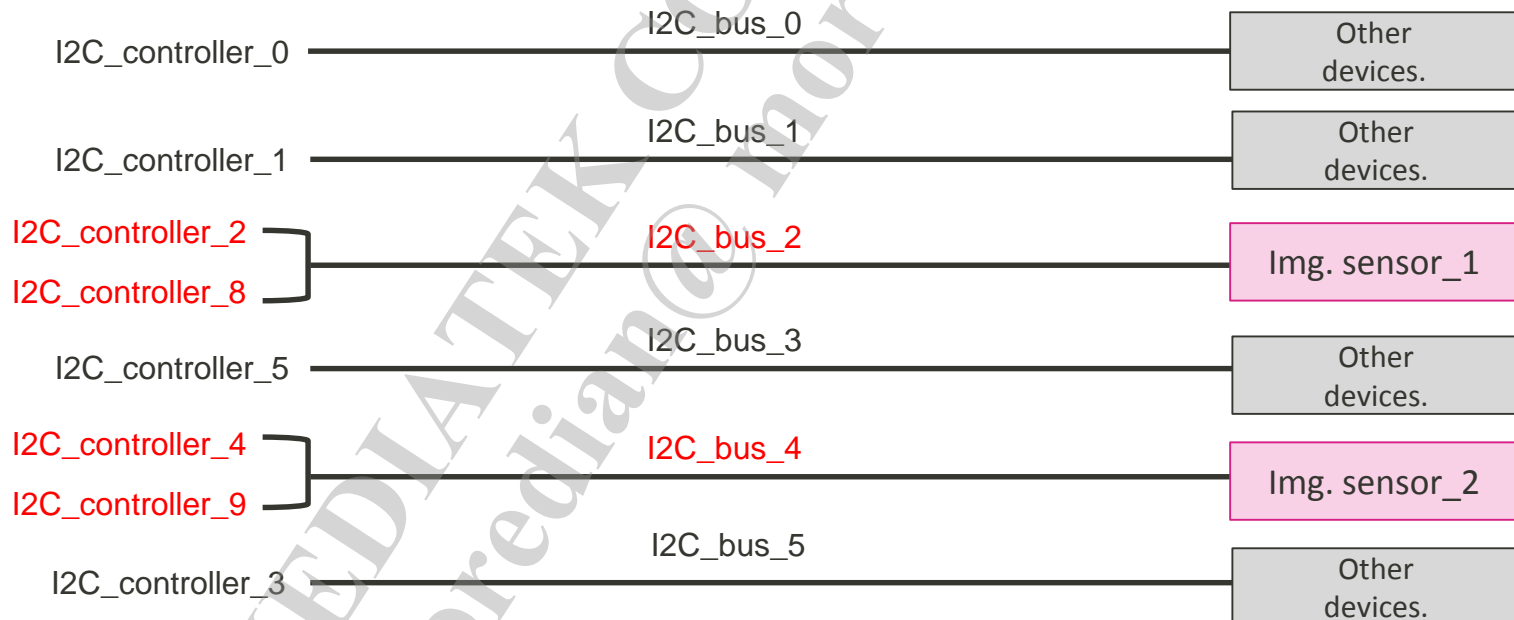
当main sensor 挂在i2c-bus 2，CCU MAIN必须配置dws为channel 8.

当main sensor 挂在i2c-bus 4，CCU MAIN必须配置dws为channel 9.

# I2C HW architecture

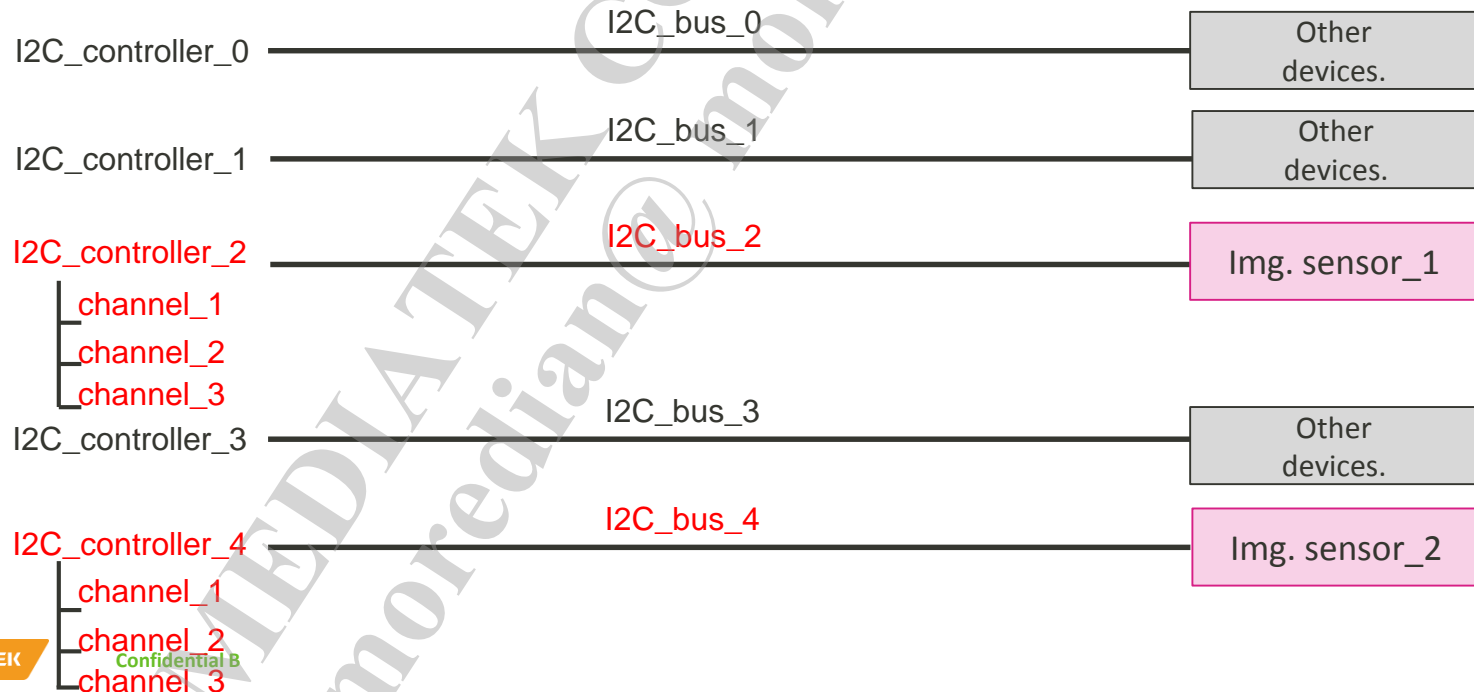
# I<sup>2</sup>C bus/controller architecture (Multi-controller version)

- There are numbers of I2C bus on chip, each bus connected with 1 controller
- But only 2 bus is connected with 2 controller
- 1 of the 2 controllers is dedicated for CCU
- Take MT6763 chip for example, I2C bus/controller topology figured as below
  - Only I2C\_bus\_2 & I2C\_bus\_4 are connected with 2 controllers
  - I2C\_controller\_8 & I2C\_controller\_9 are dedicated for CCU



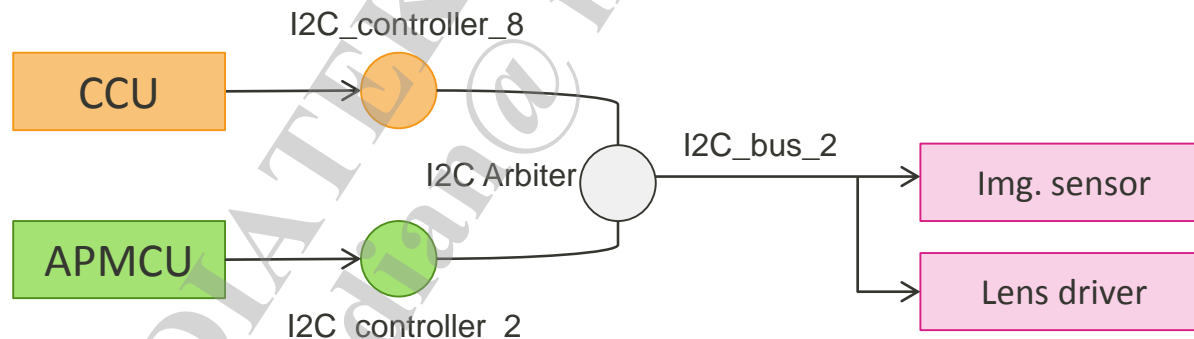
# I<sup>2</sup>C bus/controller architecture (Multi-channel version)

- There are numbers of I2C bus on chip, each bus connected with 1 controller
- I2C\_controller support multi-user by multi-channel
  - each channel with one set of control registers
- There are two I2C\_controller support multi-channel
  - In MT6771, I2C\_controller\_2 and I2C\_controller\_4 are equipped with multi-channel
  - In MT6775, I2C\_controller\_2 and I2C\_controller\_4 are equipped with multi-channel



# CCU/APMCU sensor I2C controlling scenario

- During camera working, CCU and APMCU will use same I2C bus simultaneously
  - CCU might set shutter/gain into img. Sensor
  - APMCU might read temperature from img. Sensor, and control lens via lens driver
- Since CCU & APMCU is separated core, they can't use same controller simultaneously
  - There's no cross processor SW mutex
- Each should use it's own I2C controller, the arbiter will do scheduling for I2C transactions coming from both controller
- Thus only buses with CCU-dedicated controller is feasible for img. sensor to attach on



# How-to: configure I2C controller for CCU

# How-to: configure I2C controller for CCU (1)

- Assigning I2C controller to CCU is done by configuration in **DWS file** (project dependent)
- There are 2 ways to configure DWS for CCU I2C controller mapping
- Note that DWS configuration must be **correspond to HW connection of img. sensor/I2C bus**, and make sure **HW connection meets constraint of previous page**, check constraint here
  
- **Method 1: use DrvGenTool (illustration is on next page)**
  - Execute “**vendor\mediatek\proprietary\scripts\dct\DrvGen.exe**”
  - Open file “**kernel-4.4\drivers\misc\mediatek\dws\mt6763\<project\_name>.dws**”
  - Jump to tab “**I2C**”
  - Add entry for “**CCU\_SENSOR\_I2C\_MAIN\_HW**”
    - Check the I2C HW table **link here**, choose **corresponding controller number** depend on **which I2C bus RearCAM(Main sensor) is connected to**
    - Here Main sensor is connected to “**I2C\_bus\_2**”, thus controller for CCU should be “**I2C\_controller\_8**”
    - Fill “Device Address” with any number, just make sure it differs from other entries
  - Add entry for “**CCU\_SENSOR\_I2C\_SUB\_HW**”
    - Check the I2C HW table **link here**, choose **corresponding controller number** depend on **which I2C bus frontCAM(Sub sensor) is connected to**
    - Here Main sensor is connected to “**I2C\_bus\_3**”, thus controller for CCU should be “**I2C\_controller\_9**”
    - Fill “Device Address” with any number, just make sure it differs from other entries
  - Click Save
  - Re-build kernel or full-build to make it take effect

# How-to: configure I2C controller for CCU (1)

- Illustration of Method 1: use DrvGenTool (k63v1\_64 on MT6763)

SP DrvGenTool - [MT6763::Bianco]

1 4 2 3

Projects

- MT6763
  - Bianco
    - ADC
    - ClockBuffer
    - EINT
    - GPIO
    - I2C
    - KEYPAD
    - MD1\_EINT
    - PMIC
    - POWER

ID	Speed(kbps)	Pull&Push En
BUS0	400	<input type="checkbox"/>
BUS1	400	<input type="checkbox"/>
BUS2	400	<input type="checkbox"/>
BUS3	400	<input type="checkbox"/>
BUS4	400	<input type="checkbox"/>
BUS5	3400	<input checked="" type="checkbox"/>
BUS6	3400	<input checked="" type="checkbox"/>
BUS7	400	<input type="checkbox"/>
BUS8	400	<input type="checkbox"/>
BUS9	400	<input type="checkbox"/>

ID	Slave Device	Channel	Device Address
6	EXT_BUCK_LP4	I2C_CHANNEL_1	0x57
7	EXT_BUCK_LP4X	I2C_CHANNEL_1	0x50
8	GYRO	I2C_CHANNEL_1	0x69
9	CAMERA_MAIN	I2C_CHANNEL_2	0x1A
10	CAMERA_MAIN_AF	I2C_CHANNEL_2	0x0C
11	CAMERA_MAIN_EEPROM	I2C_CHANNEL_2	0x1B
12	NFC	I2C_CHANNEL_3	0x08
13	CAMERA_SUB	I2C_CHANNEL_4	0x10
14	CAMERA_SUB_AF	I2C_CHANNEL_4	0x0C
15	CAMERA_SUB_EEPROM	I2C_CHANNEL_4	0x1C
16	MT6370_PMU	I2C_CHANNEL_5	0x34
17	USB_TYPE_C	I2C_CHANNEL_5	0x4E
18	VPROC_BUCK	I2C_CHANNEL_6	0x6B
19	CCU_SENSOR_I2C_MAIN_HW	I2C_CHANNEL_8	0x33
20	CCU_SENSOR_I2C_SUB_HW	I2C_CHANNEL_9	0x43
21	ALSPS		
22	BAROMETER		
23	BUCK_BOOST		
24	BYPASS_BOOST		
25	CAMERA_MAIN		
26	CAMERA_MAIN_AF		
27	CAMERA_MAIN_EEPROM		
28	CAMERA_MAIN_HW		
29	CAMERA_MAIN_TWO		
30	CAMERA_MAIN_TWO_AF		
31	CAMERA_SUB		
32	CAMERA_SUB_AF		
33	CAMERA_SUB_EEPROM		
34	CAP_TOUCH		
35	CCU_SENS_MAIN_HW		
36	CCU_SENS_C SUB HW		

setting | operation

Choose from dropdown list



# How-to: configure I2C controller for CCU (2)

## ■ Method 2: modify DWS file directly

- Open file “kernel-4.4\drivers\misc\mediatek\dws\mt6763\<project\_name>.dws”
- Find module I2C
- Add entry for “CCU\_SENSOR\_I2C\_MAIN\_HW”
  - Check the I2C HW table [link here](#), choose corresponding controller number depend on which I2C bus RearCAM(Main sensor) is connected to
  - If Main sensor is connected to “I2C\_bus\_2”, thus controller for CCU should be “I2C\_controller\_8”
  - Fill “Device Address” with any number, just make sure it differs from other entries
- Add entry for “CCU\_SENSOR\_I2C\_SUB\_HW”
  - Check the I2C HW table [link here](#), choose corresponding controller number depend on which I2C bus frontCAM(Sub sensor) is connected to
  - If Sub sensor is connected to “I2C\_bus\_4”, thus controller for CCU should be “I2C\_controller\_9”
  - Fill “Device Address” with any number, just make sure it differs from other entries
- Save the DWS file
- Re-build kernel or full-build to make it take effect

```
<module name="i2c">
...
<device9>
  <varName>CAMERA_MAIN</varName>
  <channel>I2C_CHANNEL_2</channel>
  <address>0x1A</address>
</device9>
...
<device13>
  <varName>CAMERA_SUB</varName>
  <channel>I2C_CHANNEL_4</channel>
  <address>0x10</address>
</device13>
...
<device18>
  <varName>VPROC_BUCK</varName>
  <channel>I2C_CHANNEL_6</channel>
  <address>0x6B</address>
</device18>
<device19>
  <varName>CCU_SENSOR_I2C_MAIN_HW</varName>
  <channel>I2C_CHANNEL_8</channel>
  <address>0x33</address>
</device19>
<device20>
  <varName>CCU_SENSOR_I2C_SUB_HW</varName>
  <channel>I2C_CHANNEL_9</channel>
  <address>0x43</address>
</device20>
</module>
```

**MEDIATEK**

*everyday genius*