

# Yocto Development Guide

## 1 Compiling Yocto Linux Firmware

Note:

- Compile environment, please use Ubuntu 22.04 (real machine or docker container),using a different version may cause compilation errors.
- Linux The SDK is cross-compiled.
- Get,compile, please use ordinary users throughout the SDK.

- a) Create project folders on Ubuntu,such as Linux \_ SDK.
- b) Install the tool

```
sudo apt update
sudo apt-get install repo git python ssh make gcc libssl-dev liblz4-tool \
expect g++ patchelf chrpath gawk texinfo chrpath diffstat binfmt-support \
gemu-user-static live-build bison flex fakeroot cmake gcc-multilib g++-multilib \
unzip device-tree-compiler ncurses-dev
```

- c) Get SDK

Initialize the warehouse::

```
repo init -u https://github.com/namtso2207/linux_manifest.git -b master -m namtso-a10-3588-
linux.xml
```

Synchronization code:

```
repo sync -j32
```

## 2 Introduction to the catalog

```
|—— Makefile -> device/rockchip/common/Makefile
|—— README.md -> device/rockchip/common/README.md
|—— app      #Stores upper-level apps, mainly Demos.
|—— build.sh -> device/rockchip/common/scripts/build.sh # Compile the script
|—— buildroot # Storage Buildroot,the root of development piece system
|—— common -> device/rockchip/common
|—— debian # Storage for Debian-developed Root File System
|—— device  # Board-level configuration of storage cores and some compilations. And
package firmware scripts and software.
|—— docs    #Documentation
|—— external # Store in the third relevant warehouse, including display, video, camera,
network and security
|—— kernel  # Storage Kernel developed code
```

```

├── output # Store the firmware information, compilation information, XML, host environment,
etc.
├── prebuilts # Store the cross-compiler chain
├── rkbin # Storage Rockchip Related Binary and tool
├── rkflash.sh -> device/rockchip/common/scripts/rkflash.sh
├── ubuntu # Storage Ubuntu the root of development file system
├── tools # Storage of common tools for Linux and Window operating systems
├── u-boot # Storage U-Boot code
├── uefi
└── yocto # Storage Yocto the root of development file system.

```

### 3 Profile Introduction

a) In the device/rockchip/rk3588/directory, there are configuration files (xxx\_defconfig) for different board types, which are used to manage the compilation configuration of each step of the SDK.

```

RK_YOCTO_MACHINE="namtso-a10-3588" # Yocto system machine configuration
RK_WIFIBT_TTY="ttyS6" # String number used by BT
RK_KERNEL_DTS_NAME="rk3588-namtso-a10-3588" #kernel dts file
RK_KERNEL_CFG="namtso-a10-3588_defconfig" # kernel configuration file
RK_UBOOT_CFG="namtso-a10-3588" # u-boot configuration file

```

b) Parameter partition table: device/rockchip/rk3588/parameter.txt

```

FIRMWARE_VER: 1.0
MACHINE_MODEL: RK3588
MACHINE_ID: 007
MANUFACTURER: RK3588
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 0xffffffff
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
GROW_ALIGN: 0
CMDLINE:
mtdparts=:0x00002000@0x00004000(uboot),0x00002000@0x00006000(misc),0x00020000@0x000
08000(boot),0x00040000@0x00028000(recovery),0x00010000@0x00068000(backup),0x01c00000@
0x00078000(rootfs),0x00040000@0x01c78000(oem),-@0x01cb8000(userdata:grow)
uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9
uuid:boot=7A3F0000-0000-446A-8000-702F00006273

```

CMDLINE attribute is our focus. Take uboot as an example, 0x00002000 @ 0x00004000 (uboot) 0x00004000 is the starting position of uboot partition, 0x00002000 is the size of partition, and so on

## 4 Compile Yocto Firmware

Fully automatic compilation:

```
./build.sh lunch # Select the configuration, and then select 2 after lunch, that is:  
namtso_a10_3588_defconfig  
export RK_ROOTFS_SYSTEM=yocto
```

The fully automated compilation generates the: output/update/Image/update.img

Compile u-boot:

```
./build.sh uboot
```

Compile kernel:

```
./build.sh kernel
```

Compile Recovery:

```
./build.sh recovery
```

Compile Yocto file system:

```
./build.sh yocto
```

Compile the rootfs.img under yocto build/lastest.

## 5 Mirror burning

### 5.1 Preparation of tools

- A10-3588
- Firmware
- Host
- Good Type-C cables

### 5.2 Entering upgrade mode

There are three ways to enter the upgrade mode, as follows:

- Serial port mode
- Linux Command Line Mode
- TST mode (recommended)

#### a) Serial Port Mode

- i. Sets the serial port.
- ii. To ensure that the serial port is connected correctly
- iii. Press the space to enter the serial port command line mode when the system is started

- iv. Enter the reboot loader to enter the burn mode

```
Hit SPACE in 0 seconds to stop autoboot
namtso#
namtso# reboot loader
```

- b) Linux Command Line Mode

Enter "reboot loader" in the Linux terminal to enter the burning mode:

```
root@rk3588-yocto:~# reboot loader
```

- c) TST mode (recommended)

- i. Energizes A10-3588.
- ii. Press the "Func" button 3 times within 2 seconds and release.
- iii. You will notice that the upgrade tool has entered upgrade mode (maskrom mode)

### 5.3 Image burning under Linux (Ubuntu) system

- a) Burn by partition:

```
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool ul rockdev/MiniLoaderAll.
bin -noreset: Burn MiniLoader
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool di -
p rockdev/parameter.txt: Burn parameter
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool di -
u rockdev/uboot.img: Burn uboot mirror image
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool di -b rockdev/boot.img:
Burn kernel mirror image
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool di -
recovery rockdev/recovery.img: Burn recovery mirror image
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool di -
rootfs rockdev/rootfs.img: Burn rootfs
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool rd: reset device
```

- b) Upgrading packaged full firmware

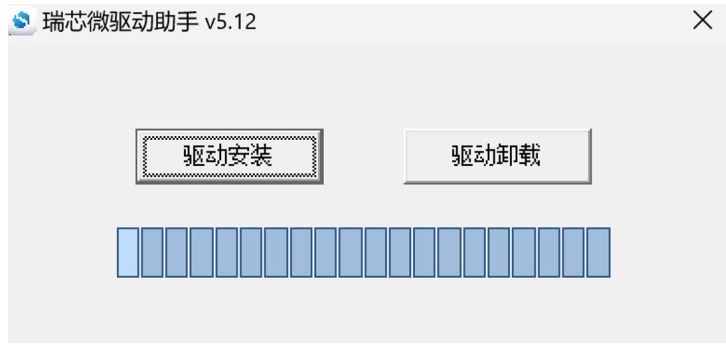
```
sudo tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool uf rockdev/update.img
```

### 5.4 Burning under Windows system

- a) Installation of programming tool

Install the RK USB driver:

- i. The USB driver DriverAssistant needs to be updated to V5.1.1.
- ii. Download driver-assitant\_v5.12.zip, unzip it, and run the DriverInstall.exe inside.
- iii. In order for all devices to use the updated driver, select driver uninstall first, and then select driver install.



b) Windows burning tool:



Click the "Firmware" button to load the firmware, and then click the "Upgrade" button.

## 6 Interface usage

### 6.1 GPIO usage

GPIO, which stands for General-Purpose Input/Output, is a general-purpose pin that can be dynamically configured and controlled during software operation. The initial state of all the GPIOs after power-on is an input mode, which can be set as a pull-up pin or a pull-down pin through software, and can also be set as an interrupt pin. The driving strength is programmable, and the core of the method is to fill the method and parameters of the GPIO bank, and to call the `gpiochip_add` to register in the kernel.

This paper takes the GPIO port of GPIO 4\_ C3 as an example to introduce the method of operating GPIO.

GPIO 4\_ C3 this GPIO is used as pwm output by default. To use it as gpio, you need to use the Device Tree Overlay function.

Enable the GPIO function as follows:

- 1) vi open file: `/boot/dtb/rockchip/rk3588-namtso-a10-3588.dtb.overlay.env`
- 2) To modify a file: `fdt_overlays=pwm-gpio-overlay`

3) Save the file, do a sync, and reboot.

### 6.1.1 GPIO pin calculation

RK3588 has 5 groups of GPIO banks: GPIO0 ~ GPIO4, and each group is numbered by A0 ~ A7, B0 ~ B7, C0 ~ C7, D0 ~ D7. The following formula is commonly used to calculate the pins:

- GPIO pin foot calculation formula:  $\text{pin} = \text{bank} * 32 + \text{number}$
- GPIO group number calculation formula:  $\text{number} = \text{group} * 8 + X$

The follow illustrates that calculation of the GPIO 4\_ C3 pin:

- $\text{bank} = 4;$  //GPIO4\_C3=>4, bank  $\in [0,4]$
- $\text{group} = 2;$  //GPIO4\_C3=>2, group  $\in \{(A=0), (B=1), (C=2), (D=3)\}$
- $X = 3;$  //GPIO4\_C3=>3, X  $\in [0,7]$
- $\text{number} = \text{group} * 8 + X = 2 * 8 + 3 = 19$
- $\text{pin} = \text{bank} * 32 + \text{number} = 4 * 32 + 19 = 147$

### 6.1.2 Input and output

a) GPIO Using GPIOs in user space

When the GPIO 4\_C3 pin is not multiplexed by other peripherals, we can export the pin to use it.

```
echo 147 > /sys/class/gpio/export
```

In the following example, GPIO 4\_C3 is set as an output and the output is high:

```
echo out > /sys/class/gpio/gpio147/direction
echo 1 > /sys/class/gpio/gpio147/value
```

Output Low:

```
echo 0 > /sys/class/gpio/gpio147/value
```

b) Using GPIO in the Kernel

Device Tree add node: kernel/arch/arm64/boot/dts/rockchip/rk3588-namtso-a10-3588.dts

```
normal gpio:
/{
.....
dts config:
gpio_demo: gpio_demo{
compatible = "namtso,a10-rk3588-gpio";
status = "okay";
pinctrl-names = "default";
pinctrl-0 = <&pin147_gpio >;
namtso-gpio = <&gpio4 RK_PC3 GPIO_ACTIVE_HIGH>;
```

```

};
.....
}

&pinctrl {
.....
gpio{
pin147_gpio: pin147-gpio{
rockchip,pins =
<4 RK_PC3 0 &pcfg_pull_none>;
};
};
.....
};

```

GPIO\_ACTIVE\_HIGH indicates active high. If you want active low, change to GPIO\_ACTIVE\_LOW. This property is read by the driver.

Then, the resource added by DTS is parsed in the probe function, and the code is as follows:

```

static int namtso_gpio_demo_probe(struct platform_device * pdev)
{
int ret = -1;
int gpio = -1;
enum of_gpio_flags flag;
struct device_node * namtso_gpio_node = pdev->dev.of_node;
struct namtso_gpio * namtso_gpio_info = devm_kzalloc(&pdev-
>dev, sizeof(struct namtso_gpio), GFP_KERNEL);
if (!namtso_gpio_info) {
pr_err("devm_kzalloc failed");
return -ENOMEM;
}

gpio = of_get_named_gpio_flags(namtso_gpio_node, "namtso-gpio", 0, &flag);
if (!gpio_is_valid(gpio)) {
pr_err("gpio: %d is invalid\n", gpio);
return -ENODEV;
}

if (gpio_request(gpio, "namtso-gpio")) {
pr_err("gpio %d request failed!\n", gpio);
return -ENODEV;
}
}

```

```

namtso_gpio_info->gpio = gpio;
namtso_gpio_info->gpio_value = (flag == OF_GPIO_ACTIVE_LOW) ? 0:1;
gpio_direction_output(namtso_gpio_info->gpio, namtso_gpio_info->gpio_value);
platform_set_drvdata(pdev, namtso_gpio_info);
pr_info("%s probe finished\n", __func__);
return 0;
}

```

The `of_get_named_gpio_flags` reads the `namtso-gpio` GPIO configuration number and flag from the device tree, the `gpio_is_valid` judges whether the GPIO number is valid, and the `gpio_request` applies to occupy the GPIO. If the initialization process fails, the `gpio_free` needs to be called to release the previously requested and successful GPIO. You can set whether to output high or low level by calling the `gpio_direction_output` in the drive. The default output here is the effective level `GPIO_ACTIVE_HIGH` obtained from DTS, that is, high level. If the drive works normally, you can use a multimeter to measure that the corresponding pin should be high level. In practice, if you want to read out the GPIO, you need to set it to the input mode first, and then read the value:

```

int val;
gpio_direction_input(your_gpio);
val = gpio_get_value(your_gpio);

```

The following are common GPIO API definitions:

```
#include <linux/gpio.h>
```

```
#include <linux/of_gpio.h>
```

```

int of_get_named_gpio_flags(struct device_node *np, const char *propname,
int index, enum of_gpio_flags *flags);
int gpio_is_valid(int gpio);
int gpio_request(unsigned gpio, const char *label);
void gpio_free(unsigned gpio);
int gpio_direction_input(int gpio);
int gpio_direction_output(int gpio, int v);

```

### 6.1.3 Interruptions

The interrupt usage of the GPIO port is similar to the input and output of the GPIO.

Device Tree add node:: kernel/arch/arm64/boot/dts/rockchip/rk3588-namtso-a10-3588.dts



```
gpio {
    compatible = "namtso,rk3588-gpio";
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&pin147_gpio >;
    namtso-irq-gpio = <&gpio4 RK_PD3 IRQ_TYPE_EDGE_RISING>;
};
```

The IRQ\_TYPE\_EDGE\_RISING indicates that the interrupt is triggered by a rising edge, and the interrupt function can be triggered when a rising edge signal is received on this pin. It can also be configured as follows:

```
IRQ_TYPE_NONE // Default, no interrupt trigger type defined
IRQ_TYPE_EDGE_RISING // Rising edge trigger
IRQ_TYPE_EDGE_FALLING // Falling edge trigger
IRQ_TYPE_EDGE_BOTH // Rising edge and trigger on both falling ed
IRQ_TYPE_LEVEL_HIGH // High level trigger
IRQ_TYPE_LEVEL_LOW // Low level trigger
```

Then, the resource added by DTS is parsed in the probe function, and then the interrupt registration application is made. The code is as follows:

```
static int namtso_gpio_demo_probe(struct platform_device * pdev)
{
    int ret = -1;
    int gpio = -1;
    enum of_gpio_flags flag;
    struct device_node * namtso_gpio_node = pdev->dev.of_node;
    struct namtso_gpio * namtso_gpio_info = devm_kzalloc(&pdev->dev, sizeof(struct namtso_gpio), GFP_KERNEL);
    if (!namtso_gpio_info) {
        pr_err("devm_kzalloc failed");
        return -ENOMEM;
    }

    gpio = of_get_named_gpio_flags(namtso_gpio_node, "namtso-gpio", 0, &flag);
    if (!gpio_is_valid(gpio)) {
        pr_err("gpio: %d is invalid\n", gpio);
        return -ENODEV;
    }

    if (gpio_request(gpio, "namtso-gpio")) {
        pr_err("gpio %d request failed!\n", gpio);
    }
}
```

```

return -ENODEV;
}
namtso_gpio_info->gpio = gpio;
namtso_gpio_info->gpio_value = (flag == OF_GPIO_ACTIVE_LOW) ? 0:1;
//gpio_direction_output(namtso_gpio_info->gpio, namtso_gpio_info->gpio_value);

namtso_gpio_info->gpio_irq = gpio_to_irq(namtso_gpio_info->gpio);
if (namtso_gpio_info->gpio_irq) {
    ret = request_irq(namtso_gpio_info->gpio_irq, namtso_gpio_irq_isr, IRQF_TRIGGER_RISING | IRQF_TRIGGER_FALLING, "namtso_gpio_irq", namtso_gpio_info);
    {
        gpio_free(namtso_gpio_info->gpio);
        dev_err(&pdev->dev, "Failed to request IRQ: %d\n", ret);
    }
}
platform_set_drvdata(pdev, namtso_gpio_info);
pr_info("%s probe finished\n", __func__);
return 0;
}

```

### 6.1.4 GPIO multiplexing

GPIO4\_C3 can be multiplexed into the following 6 functions:

- I2C7\_SCL\_M1
- PWM4\_M1
- SPI3\_CS1\_M0
- I2S2\_SDO\_M0
- GMAC0\_MCLKINOUT
- GPIO4\_C3

By default, our SDK release GPIO 4\_C3 as that pwm pin.

### 6.1.5 Commissioning method

View gpio usage:

```
cat /sys/kernel/debug/gpio
```

View pinmux-pins:

```
cat /sys/kernel/debug/pinctrl/pinctrl-rockchip-pinctrl/pinmux-pins
```

## 6.2 Use of PWM

Take pwm4 as an example here.

### 6.2.1 pwm configure

a) To enable pwm4, the following configuration is required, which is configured by default.

```
&pwm4 {
    pinctrl-0 = <&pwm4m1_pins>;
    status = "okay";
};
```

The pin corresponding to pwm4m1 is the GPIO 4\_C3. After enabling the above nodes, pwmchipx will be generated under /sys/class/pwm.

b) How do i confirm which pwmchip?

```
cat /sys/class/pwm/pwmchip*/device/uevent
```

c) Look at the OF\_FULLNAME value,

The full node for pwm4 is pwm@febd0000. The OF\_FULLNAME value in /sys/class/pwm/pwmchip0/device/uevent is also pwm@febd0000. So pwmchip0 corresponds to pwm4.

### 6.2.2 Process for using pwm

In the pwmchipx record, write 0 to the export file, that is, turn on the pwm timer 0, and a pwm0 record will be generated. The record contains the following files:

- enable: Write 1 to enable pwm and write 0 to close pwm;
- polarity: There are two parameters, normal or inversed, and the level of the output pin of the meter is inverted;
- duty\_cycle: In normal mode, the duration of the level in one cycle (unit: ns); in reversed mode, the duration of the low level in one cycle (unit: ns);
- period: The period of the pwm wave (unit: nanosecond);
- oneshot\_count: Table number of pwm waveforms in one-shot mode, the value cannot exceed 255;

Set pwm0 as follows; output frequency 100K, duty ratio 50%, positive polarity, continuous mode output:

```
cd /sys/class/pwm/pwmchip0/
echo 0 > export
cd pwm0
echo 10000 > period
echo 5000 > duty_cycle
echo normal > polarity
echo 1 > enable
```

### 6.2.3 pwm using pwm in the Kernel

```

/{
.....
  pwm_demo: pwm_demo {
    compatible = "namtso,rk3588-pwm-demo";
    status = "okay";
    pwms = <&pwm4 0 10000000 1>;
    duty_ns = <5000000>;
  };
.....
};

&pwm4 {
  pinctrl-0 = <&pwm4m1_pins>;
  status = "okay";
};

```

In `pwms=<&pwm4 0 10000000 1>`, `& pwm4` is defined in `rk3588s.dtsi`, parameter 0, table index (per-chip index of the PWM to request), generally 0, because Rockchip PWM each chip has only; 10000000: one cycle time is 10000000 nanoseconds, and one second has 100 10000000 nanoseconds, so the pwm output cycle is 100 Hz. 1: indicates polarity, 0 indicates normal polarity, and 1 indicates reversed polarity; `duty_ns` represents the duty cycle activation time, also in nanoseconds.

### 6.2.4 Interface description and use

You can use the PWM nodes generated in the above steps in other driver files. The specific method is as follows

i. Include the following header files in the driver:

```
#include <linux/pwm.h>
```

ii. : Apply for PWM interface:

```
struct pwm_device *devm_pwm_get(struct device *dev, const char *con_id);
```

For example:

```
struct pwm_device * pwm_demo = devm_pwm_get(&pdev->dev, NULL);
```

iii. Configure the PWM interface:

```
int pwm_config(struct pwm_device *pwm, int duty_ns, int period_ns);
```

For example:

```
pwm_config(pwm_demo, 5000000, 10000000);
```

iv. Enable PWM function interface:

```
int pwm_enable(struct pwm_device *pwm);
```

For example:

```
pwm_enable(pwm_demo);
```

v. Disable PWM interface:

```
void pwm_disable(struct pwm_device *pwm);
```

For example:

```
pwm_disable(pwm_demo);
```

vi. Release the applied PWM interface:

```
void pwm_free(struct pwm_device *pwm);
```

For example:

```
pwm_free(pwm_demo);
```

vii. Debugging method:

Find the corresponding node:

```
ls /sys/kernel/debug/*.pwm
```

View debugging information:

```
cat *.pwm
```

## 6.3 Use of Ethernet

a) Configuration of mainboard network card:

```

&gmac1 {
    /* Use rgmii-rxid mode to disable rx delay inside Soc */
    phy-mode = "rgmii-rxid";
    clock_in_out = "output";
    /*
    snps,reset-gpio = <&gpio3 RK_PB7 GPIO_ACTIVE_HIGH>;
    snps,reset-active-low; */
    /* Reset time is 20ms, 100ms for rtl8211f
    snps,reset-delays-us = <0 20000 100000>; */
    wolirq-gpio = <&gpio0 RK_PC2 GPIO_ACTIVE_LOW>;
    wolctrl-gpio = <&gpio0 RK_PC5 GPIO_ACTIVE_LOW>;

    pinctrl-names = "default";
    pinctrl-0 = <&gmac1_miim
        &gmac1_tx_bus2
        &gmac1_rx_bus2
        &gmac1_rgmii_clk
        &gmac1_rgmii_bus
        &wol_gpio>;

    tx_delay = <0x39>;
    /* rx_delay = <0x3f>; */
    phy-handle = <&rgmii_phy>;
    status = "okay";
};

&mdio1 {
    rgmii_phy: phy@1 {
        compatible = "ethernet-phy-ieee802.3-c22";
        reg = <0x1>;
    };
};

```

b) How to use three Ethernet ports:

eth0: corresponds to the network port of the motherboard

If the N-LINK board is connected:

eth1: corresponding to 2.5G PCIE network port of N-LINK board

eth2: corresponding to 2.5g USB network port of N-LINK board

c) Connectivity test

eth0: ping -I eth0 [www.baidu.com](http://www.baidu.com)

If the N-LINK board is connected:

eth1: ping -I eth1 [www.baidu.com](http://www.baidu.com)

```
eth2: ping -I eth2 www.baidu.com
```

If you connect multiple network cables at the same time and find that IP has been obtained and you cannot ping the external network, please add the routing table for eth0/eth1/eth2:

```
route add default gw 192.168.31.1 enP2p33s0
```

## 6.4 Use of Display

### 6.4.1 Introduction to Display Port

The RK3588 has four Video output ports, each of which is bound with a fixed display controller. For example, Port0 can be used to connect with display controllers such as DP0, DP1, HDMI/eDP0 and HDMI/eDP1, and so on for other Portx.

Each Portx has its own maximum output resolution:

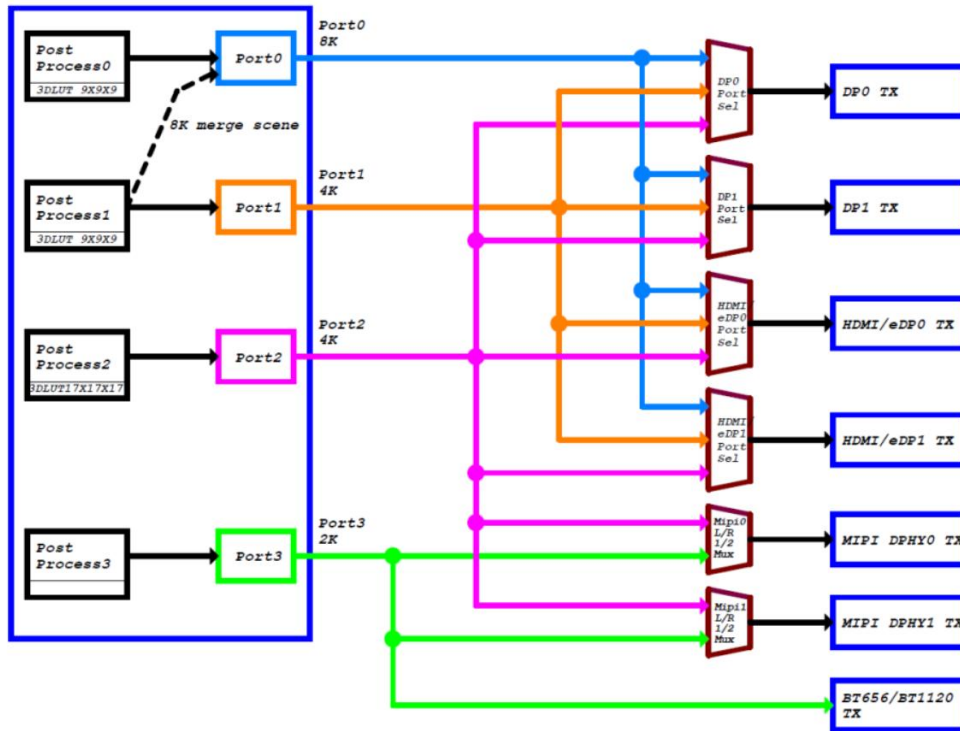
- Port 0 can output 7680x4320 @ 60Hz at most
- Port1 can output 4096x2304 @ 60Hz at most
- Port2 can output 4096x2304 @ 60Hz at most
- Port3 can output up to 1920x1080 @ 60Hz

In terms of software, if the HDMI0/1 display controller is connected to Port0 and the hardware phy supports HDMI2.1, then HDMI0/1 supports 8K @ 60Hz output.

If a separate display controller is assigned to each Portx, it can support simultaneous display (different display) of four screens at the same time.

Caution:

- When the RK3588 performs 8K output, the resources of Port0 and Port1 are actually used inside the chip at the same time, and only the Port0 port is used for output, so when the display controller connected to Port0 is connected to an external 8K display for output, abnormal operation of the display controller connected to Port1.
- Currently, one Portx can only output one resolution format at the same time, so if the software is configured with multiple display interfaces with different resolutions connected to the same Portx, only one of them can be used at the same time.



### 6.4.2 Configuration of A10-3588 display interface

The A10-3588 motherboard has four display output interfaces: HDMI2.1, DP0, MIPI1 and MIPI2. Linux system can achieve multi-screen different display. The A10-3588 N-LINK board has two display output interfaces: DP1 and EDP.

#### a) HDMI configuration up to 8K

```
&hdmi1 {
    enable-gpios = <&gpio4 RK_PB1 GPIO_ACTIVE_HIGH>;
    status = "okay";
};

&hdmi1_in_vp0 {
    status = "okay";
};

&hdmi1_sound {
    status = "okay";
};

&hdptxphy_hdmi1 {
    status = "okay";
};

&route_hdmi1 {
    status = "okay";
    connect = <&vp0_out_hdmi1>;
};
```



```
&i2s6_8ch {
    status = "okay";
};
```

By default, HDMI1 is connected to Port0, that is, to support 8K output.

**b) Up to 4K in DP0 configuration**

```
&dp0 {
    status = "okay";
};

&dp0_in_vp1 {
    status = "okay";
};

&dp0_sound{
    status = "okay";
};

&spdif_tx2 {
    status = "okay";
};
```

DP0 depends on USB-C configuration. Refer to relevant configuration of usbc0: fusb302@22, DP0 connected to VP1.

**c) DSI0 configuration**

Compatible with our TS050 and TS101 screens, DSI0 is connected to VP3.

```
vcc3v3_lcd1_en: vcc3v3-lcd1-en {
    compatible = "regulator-fixed";
    regulator-name = "vcc3v3_lcd1_en";
    regulator-min-microvolt = <3300000>;
    regulator-max-microvolt = <3300000>;
    gpio = <&gpio3 RK_PC0 GPIO_ACTIVE_HIGH>;
    enable-active-high;
    regulator-boot-on;
    regulator-state-mem {
        regulator-off-in-suspend;
    };
};

&backlight_mipi0 {
    pwms = <&pwm13 0 25000 0>;
```

```

power-supply = <&vcc3v3_lcd1_en>;
status = "okay";
};
&pwm13 {
pinctrl-0 = <&pwm13m0_pins>;
status = "okay";
};
&dsi0 {
status = "okay";
reset-delay-ms = <20>;
reset-gpios = <&gpio3 RK_PC1 GPIO_ACTIVE_HIGH>;
pinctrl-names = "default";
pinctrl-0 = <&lcd1_rst_gpio>;
};

&dsi0_panel {
status = "okay";
power-supply = <&vcc3v3_lcd1_en>;
};

&dsi0_in_vp2 {
status = "disabled";
};
&dsi0_in_vp3 {
status = "okay";
};
&route_dsi0 {
status = "okay";
connect = <&vp3_out_dsi0>;
};
&mipi_dcphy0 {
status = "okay";
};

```

#### d) DSI1 configuration

Compatible with our TS050 and TS101 screens, DSI1 is connected to VP2

```

vcc3v3_lcd2_en: vcc3v3-lcd2-en {
compatible = "regulator-fixed";
regulator-name = "vcc3v3_lcd2_en";
regulator-min-microvolt = <3300000>;
regulator-max-microvolt = <3300000>;
gpio = <&gpio3 RK_PD2 GPIO_ACTIVE_HIGH>;
enable-active-high;
regulator-boot-on;

```

```
regulator-state-mem {
    regulator-off-in-suspend;
};

};

&backlight_mipi1 {
    pwms = <&pwm10 0 25000 0>;
    power-supply = <&vcc3v3_lcd2_en>;
    status = "okay";
};

&pwm10 {
    pinctrl-0 = <&pwm10m2_pins>;
    status = "okay";
};

&dsi1 {
    reset-delay-ms = <20>;
    reset-gpios = <&gpio3 RK_PD4 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&lcd2_rst_gpio>;
    status = "okay";
};

&dsi1_panel {
    status = "okay";
    power-supply = <&vcc3v3_lcd2_en>;
};

&mipi_dcphy1 {
    status = "okay";
};

&dsi1_in_vp2 {
    status = "okay";
};

&dsi1_in_vp3 {
    status = "disabled";
};

&route_dsi1 {
    status = "okay";
    connect = <&vp2_out_dsi1>;
};
```

e) N-LINK DP1 configuration,

DP1 is also connected to VP2, cannot be used with DSI1 at the same time, and supports up to 4K

```
&dp1 {
    pinctrl-names = "default";
    pinctrl-0 = <&dp1_hpd>;
    hpd-gpios = <&gpio3 RK_PD5 GPIO_ACTIVE_HIGH>;
    status = "okay";
};

&dp1_in_vp2 {
    status = "okay";
};

&dp1_sound{
    status = "okay";
};

&spdif_tx5 {
    status = "okay";
};

&usbdp_phy1 {
    rockchip,dp-lane-mux = <0 1 2 3>;
};
```

f) N-LINK EDP configuration

The EDP screen is also connected to VP1 and cannot be used with DP0.

```
vcc3v3_edp0: vcc3v3-edp0 {
    compatible = "regulator-fixed";
    regulator-name = "vcc3v3_edp0";
    regulator-boot-on;
    regulator-always-on;
    enable-active-high;
    gpio = <&gpio3 RK_PB7 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&vcc3vc_edp0_gpio>;
};

&backlight_edp0 {
    pwms = <&pwm6 0 25000 0>;
    status = "okay";
};
```

```

&edp0 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&edp0_hpd>;
    hpd-gpios = <&gpio1 RK_PA5 GPIO_ACTIVE_HIGH>;

    ports {
        port@1 {
            reg = <1>;
            edp0_out_panel: endpoint {
                remote-endpoint = <&panel_in_edp0>;
            };
        };
    };
};

&edp0_in_vp1 {
    status = "okay";
};

&route_edp0 {
    connect = <&vp1_out_edp0>;
    status = "okay";
};

&hdptxphy0 {
    status = "okay";
};

/* edp backlight pwm */
&pwm6 {
    pinctrl-0 = <&pwm6m1_pins>;
    status = "okay";
};

```

**g) =Commissioning means**

Get information about the Video Portx (and connected display controllers) in use on the system

```
cat /sys/kernel/debug/dri/0/summary
```

**6.5 ADC usage**

There are two types of AD interfaces on the A10-3588: Temperature Sensor and Successive approximation ADC (Successive Approximation Register).

TS-ADC(Temperature Sensor): support seven channels.

SAR-ADC(Successive Approximation Register): supports eight channels of single-ended, 12-bit SAR-ADC with a maximum conversion rate of 1 MSPS and 20 MHz A/D converter clock.

The core uses an industrial I/O subsystem to control the ADC, which is mainly designed for AD or DA

conversion sensors. The following takes SAR-ADC as an example to introduce the use and configuration method of ADC.

## a) Obtaining ADC values for all channels

```
cat /sys/bus/iio/devices/iio\:device0/in_voltage*_raw
```

## b) Using ADC in the Kernel

i. To configure a DTS node:

```
adc_demo: adc_demo{
    compatible = "namtso,rk3588-adc";
    status = "okay";
    io-channels = <&saradc 3>;
};
```

ii. Driver:

```
static void namtso_adc_iio_poll(struct work_struct *work)
{
    int ret = -1;
    int value = -1;
    if (g_ctrl->chan)
    {
        ret = iio_read_channel_raw(g_ctrl->chan, &value);
        if (ret < 0)
        {
        }
    }
    mutex_lock(&g_ctrl->mutex_rd);
    g_ctrl->read_value = value;
    mutex_unlock(&g_ctrl->mutex_rd);
    schedule_delayed_work(&g_ctrl->adc_poll_work, NAMTSO_ADC_SAMPLE_JIFFIES);
}

static ssize_t namtso_adc_write(struct file *file,
    const char __user *buf, size_t n, loff_t *offset)
{
    return 0;
}

static ssize_t namtso_adc_read(struct file *file,
    char __user *buf, size_t n, loff_t *offset)
{
    int ret = -1;
    mutex_lock(&g_ctrl->mutex_rd);
```

```
ret = copy_to_user(buf, &g_ctrl->read_value, sizeof(int));
mutex_unlock(&g_ctrl->mutex_rd);
return sizeof(int);
}
static loff_t namtso_adc_lseek(struct file * file, loff_t offset, int whence)
{
return 0;
}

static int namtso_adc_open(struct inode * inode, struct file * file)
{
return 0;
}
static int namtso_adc_close(struct inode * inode, struct file * file)
{
return 0;
}
static long namtso_adc_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
{
switch(cmd)
{
case ADC_SET:
if (arg) {
schedule_delayed_work(&g_ctrl->adc_poll_work, 1000);
}
else
{
cancel_delayed_work_sync(&g_ctrl->adc_poll_work);
}
break;
default:
printk(KERN_ERR "cmd is error\n");
break;
}
return 0;
}

static const struct file_operations namtso_adc_fops = {
.write = namtso_adc_write,
.read = namtso_adc_read,
.llseek = namtso_adc_lseek,
.open = namtso_adc_open,
.release = namtso_adc_close,
.unlocked_ioctl = namtso_adc_ioctl,
.owner = THIS_MODULE,
```

```
};

static struct miscdevice namtso_adc_misc = {
    .minor = MISC_DYNAMIC_MINOR,
    .name = "namtso_adc",
    .fops = &namtso_adc_fops,
};

static int namtso_adc_iio_probe(struct platform_device *pdev)
{
    pr_info("namtso_adc_iio_probe!\n");

    g_ctrl = devm_kzalloc(&pdev->dev, sizeof(struct adc_ctrl), GFP_KERNEL);
    if (!g_ctrl)
    {
        dev_err(&pdev->dev, "devm_kzalloc fail\n");
        return -ENOMEM;
    }
    g_ctrl->chan = iio_channel_get(&pdev->dev, NULL);
    if (!g_ctrl->chan)
    {
        dev_err(&pdev->dev, "iio_channel_get fail\n");
        return -EINVAL;
    }
    mutex_init(&g_ctrl->mutex_rd);
    INIT_DELAYED_WORK(&g_ctrl->adc_poll_work, namtso_adc_iio_poll);

    misc_register(&namtso_adc_misc);
    return 0;
}

static int namtso_adc_iio_remove(struct platform_device *pdev)
{
    pr_info("namtso_adc_iio_remove!\n");
    cancel_delayed_work_sync(&g_ctrl->adc_poll_work);
    if (g_ctrl->chan)
    {
        iio_channel_release(g_ctrl->chan);
        g_ctrl->chan = NULL;
    }
    misc_deregister(&namtso_adc_misc);
    return 0;
}
```



iii. Description of drive:

Get iio channel:

```
struct iio_channel *chan; # Define IIO channel structure
g_ctrl->chan = iio_channel_get(&pdev->dev, NULL); # Get IIO channel structure
```

Read the raw data collected by AD:

```
iio_read_channel_raw(g_ctrl->chan, &value);
```

Calculate the collected voltage:

The AD converted value is converted into a voltage value required by the user using the standard voltage.

The calculation formula is as follows:

$$V_{ref} / (2^{n-1}) = V_{result} / raw$$

Formula description:

- Vref is the standard voltage
- n is the number of bits for AD conversion
- Vresult is the acquisition voltage required by the user
- raw refers to the raw data collected by AD

For example, if the standard voltage is 1.8 V, the number of AD acquisition bits is 12, and the original data acquired by AD is 1000, then:

$$V_{result} = (1800mv * 1000) / 4095;$$

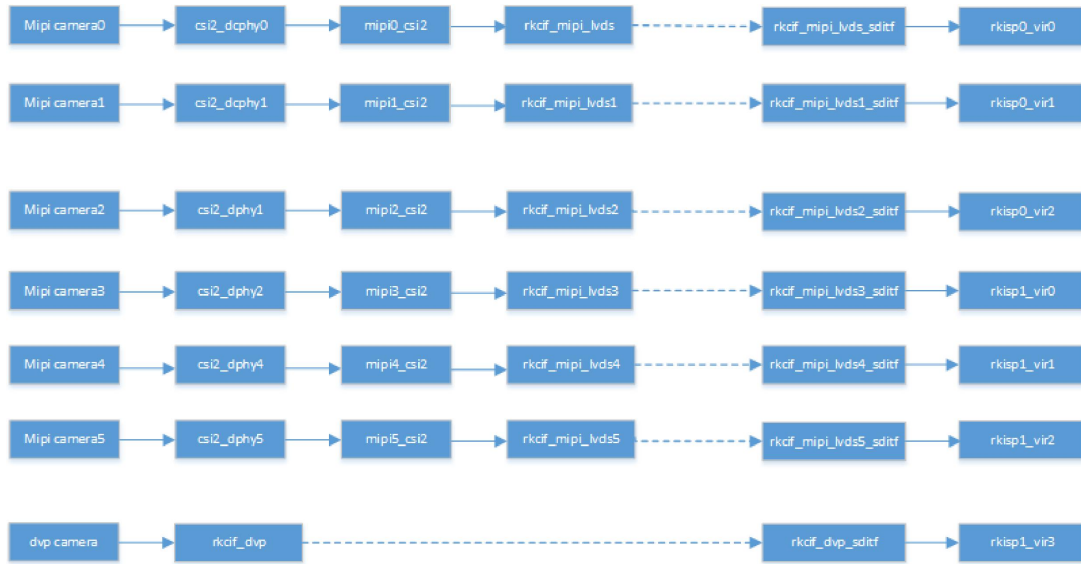
Release the channel obtained by the iio\_channel\_get function:

```
iio_channel_release(g_ctrl->chan);
```

## 6.6 Camera usage

### 6.6.1 Configuration

The RK3588 hardware supports the acquisition of up to 7 sensors: 6mipi + 1dvp, and the software channels for multiple sensors are as follows:



The rk3588 supports two dcpHys with node names csi2\_dphy0/csi2\_dphy1. Each dcpHy hardware supports the simultaneous use of RX/TX, and RX is used for camera input. Support DPHY/CPHY protocol multiplexing. It should be noted that TX/RX of the same dcpHy can only use DPHY or CPHY at the same time. Consult the rk3588 data sheet for additional dcpHy parameters.

The rk3588 supports two dphy hardwareS, here we call them dphy0\_hw/dphy1\_hw, and the two dphy hardwareS can work in both full mode and split mode.

i. dphy0\_hw

- full mode: the node name uses the csi2\_dphy0 and supports up to 4 lanes.
- split mode: split into 2 Phys, which are csi2\_dphy1 (using 0/1 lane) and csi2\_dphy2 (using 2/3 lane).

Each phy supports 2 lanes at most.

ii. dphy1\_hw

- full mode: the node name uses the csi2\_dphy3 and supports up to 4 lanes.
- split mode: split into 2 Phys, which are csi2\_dphy4 (using 0/1 lane),the csi2\_dphy5 (using 2/3 lane) and supports up to 2 lanes per phy.

To use the above mipi phy node, the corresponding physical node needs to be configured.

(csi2\_dcpHy0\_hw/csi2\_dcpHy1\_hw/csi2\_dphy0\_hw/csi2\_dphy1\_hw)

Each mipi phy needs a csi2 module to resolve the mipi protocol, and the node names are respectively mipi0\_csi2~mipi5\_csi2;

rk3588 all camera data needs to go through vicap and then link to isp. The rk3588 only supports one vicap hardware, and this vicap supports the simultaneous input of six mipi phys and one dvp data,so we divide the vicap into seven nodes such as the rkcf\_mipi\_lvds~rkcf\_mipi\_lvds5 and the rkcf\_dvp.The link relationship between each

vicap node and the isp indicates the link relationship through the corresponding virtual XXX\_sdtf.

The rk3588 supports two isp hardware, and each isp device can virtualize multiple virtual nodes. The software reads the image data of each channel from the ddr in turn for isp processing by means of readback. For a multi-shot scheme, it is recommended to distribute the data stream equally over the two isps.

#### Pass-through and Readback Mode

- Pass-through: means that the data is collected by the vicap and sent directly to the isp for processing instead of being stored in the ddr. It should be noted that in the case of hdr pass-through, only the short frame is a true pass-through, and the long frame needs to exist in the ddr, and the isp reads from the ddr.
- Readback: the data is collected to ddr through vicap. After the data is obtained, the application pushes the buffer address to isp, and isp obtains the image data from ddr.

When configured with dts an isp hardware uses the pass-through mode by default if only one virtual node is configured, and uses the readback mode by default if multiple virtual nodes are configured.

Namtso-A10 supports up to 4 4Lane cameras (IMX415/OS08A10), Two dcphys and two dphys (full mode) are used. Configuration path such as: kernel/arch/arm64/boot/dts/rockchip/rk3588-namtso-a10-3588-camera.dtsi.

As shown. The following is an example of a camera mounted on the i2c2:

```
&i2c2 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&i2c2m4_xfer>;

    dw9714_cam1: dw9714_cam1@c {
        compatible = "dongwoon,dw9714";
        status = "okay";
        reg = <0x0c>;
        pinctrl-names = "default";
        pinctrl-0 = <&focus_cam1_gpio>;
        focus-gpios = <&gpio1 RK_PB1 GPIO_ACTIVE_HIGH>;
        rockchip,vcm-start-current = <20>;
        rockchip,vcm-rated-current = <76>;
        rockchip,vcm-step-mode = <0>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "back";
    };
    imx415_cam1: imx415f@1a {
        compatible = "sony,imx415";
```

```

status = "okay";
reg = <0x1a>;
clocks = <&cru CLK_MIPI_CAMARAOUT_M1>;
clock-names = "xvclk";
power-domains = <&power RK3588_PD_VI>;
pinctrl-names = "default";
pinctrl-0 = <&mipim0_camera1_clk>, <&cam1_reset_gpio>;
rockchip,grf = <&sys_grf>;
reset-gpios = <&gpio1 RK_PB0 GPIO_ACTIVE_LOW>;
pwn-gpios = <&gpio1 RK_PB1 GPIO_ACTIVE_HIGH>;
rockchip,camera-module-index = <0>;
rockchip,camera-module-facing = "back";
rockchip,camera-module-name = "CMK-OT2022-PX1";
rockchip,camera-module-lens-name = "IR0147-50IRC-8M-F20";
lens-focus = <&dw9714_cam1>;
port {
    imx415_cam1_out: endpoint {
        remote-endpoint = <&mipi_in_dcphy1>;
        data-lanes = <1 2 3 4>;
    };
};

os08a10_cam1: os08a10_cam1@36 {
    compatible = "ovti,os08a10";
    status = "okay";
    reg = <0x36>;
    clocks = <&cru CLK_MIPI_CAMARAOUT_M1>;
    clock-names = "xvclk";
    power-domains = <&power RK3588_PD_VI>;
    pinctrl-names = "default";
    pinctrl-0 = <&mipim0_camera1_clk>, <&cam1_reset_gpio>;
    rockchip,grf = <&sys_grf>;
    reset-gpios = <&gpio1 RK_PB0 GPIO_ACTIVE_LOW>;
    power-gpios = <&gpio1 RK_PB1 GPIO_ACTIVE_HIGH>;
    rockchip,camera-module-index = <0>;
    rockchip,camera-module-facing = "back";
    rockchip,camera-module-name = "default";
    rockchip,camera-module-lens-name = "default";
    port {
        os08a10_cam1_out: endpoint {
            remote-endpoint = <&mipi_in_dcphy4>;
            data-lanes = <1 2 3 4>;
        };
    };
};

```

```
};

};
};

&csi2_dcphy1 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;
        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi_in_dcphy1: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&imx415_cam1_out>;
                data-lanes = <1 2 3 4>;
            };

            mipi_in_dcphy4: endpoint@2 {
                reg = <2>;
                remote-endpoint = <&os08a10_cam1_out>;
                data-lanes = <1 2 3 4>;
            };
        };
        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            csidcphy1_out: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&mipi1_csi2_input>;
            };
        };
    };
};

&mipi_dcphy1 {
    status = "okay";
};
```

```

&mipi1_csi2 {
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {
            reg = <0>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi1_csi2_input: endpoint@1 {
                reg = <1>;
                remote-endpoint = <&csidcphy1_out>;
            };
        };

        port@1 {
            reg = <1>;
            #address-cells = <1>;
            #size-cells = <0>;

            mipi1_csi2_output: endpoint@0 {
                reg = <0>;
                remote-endpoint = <&cif_mipi_in1>;
            };
        };
    };
};

&rkcif_mipi_lvds1 {
    status = "okay";

    port {
        cif_mipi_in1: endpoint {
            remote-endpoint = <&mipi1_csi2_output>;
        };
    };
};

&rkcif_mipi_lvds1_sditf {
    status = "okay";

```

```

port {
    mipi1_lvds_sditf: endpoint {
        remote-endpoint = <&isp0_vir1>;
    };
};

&rkisp0_vir1 {
    status = "okay";
    port {
        #address-cells = <1>;
        #size-cells = <0>;

        isp0_vir1: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&mipi1_lvds_sditf>;
        };
    };
};

```

Channel description: imx415\_cam1/os08a10\_cam1s-- > csi2\_dcphy1-- > mipi1\_csi2-- > rkCIF\_mipi\_lvds1-->

rkCIF\_mipi\_lvds1\_sditfs-- > rkisp0\_vir1

### 6.6.2 Camera bottom debugging

For A10-3588 snapshot: camera 1-4 in turn.

```

v4l2-ctl -d /dev/video53 --set-fmt-video=width=3840,height=2160,pixelformat=NV12 --stream-
mmap=3 --stream-skip=20 --stream-to=/picture/data53.yuv --stream-count=1 --stream-poll

v4l2-ctl -d /dev/video44 --set-fmt-video=width=3840,height=2160,pixelformat=NV12 --stream-
mmap=3 --stream-skip=20 --stream-to=/picture/data44.yuv --stream-count=1 --stream-poll

v4l2-ctl -d /dev/video62 --set-fmt-video=width=3840,height=2160,pixelformat=NV12 --stream-
mmap=3 --stream-skip=20 --stream-to=/picture/data62.yuv --stream-count=1 --stream-poll

v4l2-ctl -d /dev/video71 --set-fmt-video=width=3840,height=2160,pixelformat=NV12 --stream-
mmap=3 --stream-skip=20 --stream-to=/picture/data71.yuv --stream-count=1 --stream-poll

```

### 6.6.3 Linux System Preview Camera

a) Camera1:

```

root@rk3588-yocto:/rockchip-test/camera# cat camera_rkisp_test.sh
#!/bin/bash
#export GST_DEBUG=*:5
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/gstreamer-1.0

gst-launch-1.0          v4l2src          device=/dev/video53      !          video/x-
raw,format=NV12,width=640,height=480,framerate=30/1 ! waylandsink

Modify the corresponding node device=/dev/video53, then execute this script to preview it.

```

b) Camera2:

```

root@rk3588-yocto:/rockchip-test/camera# cat camera_rkisp_test.sh
#!/bin/bash
#export GST_DEBUG=*:5
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/gstreamer-1.0

gst-launch-1.0          v4l2src          device=/dev/video44      !          video/x-
raw,format=NV12,width=640,height=480,framerate=30/1 ! waylandsink

Modify the corresponding node device=/dev/video44, then execute this script to preview it.

```

c) Camera3:

```

root@rk3588-yocto:/rockchip-test/camera# cat camera_rkisp_test.sh
#!/bin/bash
#export GST_DEBUG=*:5
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/gstreamer-1.0

gst-launch-1.0          v4l2src          device=/dev/video62      !          video/x-
raw,format=NV12,width=640,height=480,framerate=30/1 ! waylandsink

Modify the corresponding node device=/dev/video62, then execute this script to preview it.

```

d) Camera4:

```

root@rk3588-yocto:/rockchip-test/camera# cat camera_rkisp_test.sh
#!/bin/bash
#export GST_DEBUG=*:5
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib/gstreamer-1.0

gst-launch-1.0          v4l2src          device=/dev/video71      !          video/x-
raw,format=NV12,width=640,height=480,framerate=30/1 ! waylandsink

Modify the corresponding node device=/dev/video71, then execute this script to preview it.

```

6.6.4 How to determine the video node



The video node can be identified by using `media-ctl -p -d /dev/mediax` and then finding `rkisp_mainpath`.

```
media-ctl -p -d /dev/mediax
```

## 6.7 Use of CAN

### 6.7.1 Introduction to CAN

CAN (Controller Area Network) bus is a kind of serial communication network which effectively supports distributed control or real-time control. CAN-bus is a widely used bus protocol in automobiles, which is designed as a microcontroller communication in the automotive environment.

### 6.7.2 DTS Node Configuration

```
&can1 {
    status = "okay";
    assigned-clocks = <&cru CLK_CAN1>;
    assigned-clock-rates = <200000000>;
    pinctrl-names = "default";
    pinctrl-0 = <&can1m1_pins>;
};
```

Since the system creates only one CAN device according to the above dts node, the first device created is `can0`.

### 6.7.3 Communication test

CAN communication test: use `candump` and `cansend` tools to send and receive messages:

```
# Turn off the can0 device at the transceiver end
ip link set can0 down
# Set the bit rate to 200Kbps at the transceiver.
ip link set can0 type can bitrate 200000
# Sending and receiving end open the can0 device
ip link set can0 up
# Receiving end execution candump blocking waiting message
candump can0
# Sending end execution cansend , Send a message
cansend can0 123#aabbccceeddfggss123
```

### 6.7.4 More instructions

```

ip link set canX down // Turn off the can device;
ip link set canX up // Turn on the can device;
ip -details link show canX // Display the detailed information of can device;
candump canX // Receive data sent by that can bus;
ifconfig canX down // Turn off the can device for configuration;
ip link set canX up type can bitrate 250000 // Set can baud rate;
canconfig canX bitrate +baud rate;
canconfig canX start // Start the can device;
canconfig canX ctrlmode loopback on // Loopback test
canconfig canX restart // Restart the can device;
canconfig canX stop // Stop the can device;
canecho canX // View can device bus status;
cansend canX --identifier=ID+ data // Send data;
candump canX --filter=ID: mask // Use filter to receive ID matched data

```

## 6.8 LED Usage

The A10-3588 has an onboard WHITE\_LED with an additional EXT\_LED port brought out.

- WHITE\_LED: GPIO0\_PC7
- EXT\_LED: GPIO0\_PD0

### 6.8.1 LED configuration

```

leds {
    compatible = "gpio-leds";
    pinctrl-names = "default";
    pinctrl-0 = <&white_led_gpio &ext_led_gpio>;

    white_led {
        gpios = <&gpio0 RK_PC7 GPIO_ACTIVE_HIGH>;
        label = "white_led";
        linux,default-trigger = "none";
        default-state = "on";
    };

    ext_led {
        gpios = <&gpio0 RK_PD0 GPIO_ACTIVE_HIGH>;
        label = "ext_led";
        linux,default-trigger = "none";
        default-state = "on";
    };
};

```

### 6.8.2 LED control

- WHITE\_LED:

Turn on the LED:

```
echo default-on > /sys/class/leds/white_led/trigger
```

Turn off the LED:

```
echo none > /sys/class/leds/white_led/trigger
```

Set the heartbeat effect:

```
echo heartbeat > /sys/class/leds/white_led/trigger
```

Set Timer effect:

```
echo timer > /sys/class/leds/white_led/trigger
```

There are `delay_off` and `delay_on` in the `/sys/class/leds/white_led` directory to control the blinking effect.

## b) EXT\_LED:

Turn on the LED:

```
echo default-on > /sys/class/leds/ext_led/trigger
```

Turn off the LED:

```
echo none > /sys/class/leds/ext_led/trigger
```

Set the heartbeat effect:

```
echo heartbeat > /sys/class/leds/ext_led/trigger
```

Set Timer effect:

```
echo timer > /sys/class/leds/ext_led/trigger
```

There are `delay_off` and `delay_on` in the `/sys/class/leds/ext_led` directory to control the blinking effect.

## 6.9 PCIe Usage

Corresponding relationship between available hardware resources and software pcie controller nodes and PHY nodes of RK3588 PCIe is shown in the figure below:

资源	模式	dts 节点	可用phy	内部 DMA
PCIe Gen3 x 4 lane	RC/EP	pcie3x4:pcie@fe150000	pcie30phy	是
PCIe Gen3 x 2 lane	RC only	pcie3x2:pcie@fe160000	pcie30phy	否
PCIe Gen3 x 1 lane	RC only	pcie2x110:pcie@fe170000	pcie30phy, combphy1_ps	否
PCIe Gen3 x 1 lane	RC only	pcie2x111:pcie@fe180000	pcie30phy, combphy2_psu	否
PCIe Gen3 x 1 lane	RC only	pcie2x112:pcie@fe190000	combphy0_ps	否

RK3588 has 5 PCIe controllers in total. The hardware IP is the same, but the configuration is different. One 4Lane DM mode can be used as EP, and the other 2Lane and three 1Lane controllers can only be used as RC.

The RK3588 has two types of PCIe PHYs, one of which is the pcie3.0 PHY with two Ports and four lanes, and the other is the pcie 2.0 PHY with three lanes, each of which is a 2.0 1 Lane, combined with SATA and USB.

The 4Lane of pcie3.0 PHY can be split according to actual requirements. After splitting, the corresponding controller needs to be configured reasonably. All configurations are completed in DTS, and the driver needs to be modified.

### 6.9.1 A10-3588 PCIE Configuration

```
&pcie2x110 {
    reset-gpios = <&gpio1 RK_PB4 GPIO_ACTIVE_HIGH>;
    vpcie3v3-supply = <&vcc3v3_pcie20>;
    pinctrl-names = "default";
    pinctrl-0 = <&pcie2x110_gpio>;
    rockchip,perst-inactive-ms = <500>;
    status = "okay";
};

&pcie2x112 {
    reset-gpios = <&gpio3 RK_PD1 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&pcie2x111_gpio>;
    status = "okay";
};

&pcie30phy {

    rockchip,pcie30-phymode = <PHY_MODE_PCIE_AGGREGATION>;
```

```

status = "okay";
};
&pcie3x4 {
reset-gpios = <&gpio4 RK_PB6 GPIO_ACTIVE_HIGH>;
vpcie3v3-supply = <&vcc3v3_pcie30>;
pinctrl-names = "default";
pinctrl-0 = <&pcie3x4_gpio>;
status = "okay";
};

```

A10-3588 uses pcie2x10, pcie2x12, pcie3x4 controllers.

- The pcie2x10 is connected to the PCIEX1\_0 Switch, and the WIFI/PCIE network card can be selected through the LINK/M2-PCIE\_SW.
- pcie2x12 to SATA controller.
- pcie3x4 to M2\_M-KEY

Node description:

- reset-gpios: reset pin attribute
- vcc3v3\_pcie30: power supply pin node.

### 6.9.2 WIFI/PCIE network card switching

pcie2x10 is connected to PCIE network card by default. To use WIFI function,edit /boot/uEnv.txt,wifi=off should be wifi=on,and then restart after sync to connect to wifi.

### 6.10 SATA

View the device:

```
ls /dev/sd*
```

Mount:

```
mount /dev/sd* /mnt/storage
```

Speed test:

```
cd /mnt/
iozone -e -l -a -s 100M -r 4k -r 16k -r 512k -r 1024k -r 16384k -i 0 -i 1 -i 2 iozone
```

### 6.11 M.2 SSD

i. View the device:

```
ls /dev/nvme*
```

ii. Read-write speed measurement:

Write test:

```
fio -filename=/dev/nvme0n1 -direct=1 -iodepth 4 -thread=1 -rw=write -ioengine=libaio -
bs=1M -size=200G -numjobs=30 -runtime=60 -group_reporting -name=my
```

Read test:

```
fio -filename=/dev/nvme0n1 -direct=1 -iodepth 4 -thread=1 -rw=read -ioengine=libaio -
bs=1M -size=200G -numjobs=30 -runtime=60 -group_reporting -name=my
```

## 6.12 Use of RTC

### 6.12.1 Introduction

A10-3588 development board uses PT7C4363 as RTC (Real Time Clock). It is a low-power CMOS real-time clock/calendar chip. It provides a programmable clock output, an interrupt output and a power-down detector. All addresses and data are transmitted serially through I2C bus interface. The maximum bus speed is 400Kbits/s, and the embedded word address register is automatically incremented each time data is read or written. The characteristics are as follows:

- Timable seconds, minutes, hours, weeks, days, months and years based on a 32.768 kHz crystal.
- Wide operating voltage range: 1.0~5.5V.
- Low sleep current: typical 0.25μA (VDD=3.0V, TA= 25°C).
- Internal integrated oscillation capacitor.
- Open drain interrupt pin.

Note: When using RTC, please connect the RTC battery.

### 6.12.2 RTC Configuration

```
pt7c4363: pt7c4363@51 {
    compatible = "haoyu,hym8563";
    reg = <0x51>;
    #clock-cells = <0>;
    clock-frequency = <32768>;
    clock-output-names = "pt7c4363";
    wakeup-source;
};
```

### 6.12.3 Interface Usage

Linux provides four user-space call interfaces:

- SYSFS interface: /sys/class/rtc/rtc0/
- PROCFS interface: /proc/driver/rtc
- IOCTL interface: /dev/rtc0

- hwclock command

a) SYSFS interface

View the date and time of the current RTC:

```
root@rk3588-yocto:/sys/class/rtc/rtc0# cat date
2024-04-10
root@rk3588-yocto:/sys/class/rtc/rtc0# cat time
09:30:46
root@rk3588-yocto:/sys/class/rtc/rtc0#
```

b) PROCFS interface

Print RTC related information:

```
root@rk3588-yocto:~# cat /proc/driver/rtc
rtc_time      : 09:40:22
rtc_date      : 2024-04-10
alarm_time    : 09:38:00
alarm_date    : 2024-04-11
alarm_IRQ     : no
alarm_pending : no
update IRQ enabled : no
periodic IRQ enabled : no
periodic IRQ frequency : 1
max user IRQ frequency : 64
24hr         : yes
```

c) IOCTL interface

You can use ioctl to control /dev/rtc0

d) hwclock sets the time

```
date 092115372023.30      Set system time
hwclock -r                View rtc time
hwclock -w                Sync system time is up rtc
```

### 6.13 SPI Usage

#### 6.13.1 Introduction to SPI

SPI is a high-speed, full-duplex, synchronous serial communication interface for connecting microcontrollers, sensors, storage devices, etc.

#### 6.13.2 Working mode of SPI

The SPI operates in a master-slave mode, which typically has one master and one or more slaves, requiring at least four wires:

- CS      chip select signal

- SCLK            clock signal
- MOSI           master data output, slave data input
- MISO           master data input, slave data output

The Linux kernel uses a combination of CPOL and CPHA to represent the four modes of operation of the current SPI:

- CPOL = 0, CPHA = 0            SPI\_MODE\_0
- CPOL = 0, CPHA = 1            SPI\_MODE\_1
- CPOL = 1, CPHA = 0            SPI\_MODE\_2
- CPOL = 1, CPHA = 1            SPI\_MODE\_3

CPOL: indicates the state of the initial level of the clock signal, where 0 is low and 1 is high.

CPHA: Indicates on which clock edge to sample, 0 for the first clock edge and 1 for the second.

### 6.13.3 Drive development

Take mcp2515 as an example, configure the DTS node:

Add the SPI driver node description in kernel-5.10/arch/arm64/boot/dts/rockchip/rk3588-namtso-a10-3588.dts ,as follows:

```
&spi3 {
    pinctrl-names = "default";
    pinctrl-0 = <&spi3m0_cs0 &spi3m0_pins>;
    num-cs = <1>;
    status = "okay";
    spi_can@0 {
        compatible = "microchip,mcp2515";
        reg = <0>;
        clocks = <&mcp251x_clk>;
        spi-max-frequency = <20000000>;

        status = "okay";

        status = "okay";
        interrupt-parent = <&gpio0>;
        interrupts = <RK_PC6 IRQ_TYPE_EDGE_FALLING>;
    };
};
```

- status:Set to okay if SPI is to be enabled, or disable if it is not to be enabled.
- spi\_can@0:Since CS0 is used in this example, it is set to 00 here. If CS1 is used, it is set to 01.
- compatible: The attribute here must be consistent with the member compatible in the struct in the



driver: of\_device\_id.

- reg: This is consistent with the spi\_can@0. In this example, it is set to: 0.
- spi-max-frequency: This sets the maximum frequency used by the spi. The A10-3588 supports up to 50000000.

### 6.13.4 Interface Usage

Linux provides a limited SPI user interface. If you do not need to use IRQ or other kernel-driven interfaces, you can consider using the interface spidev to write user-level programs to control SPI devices. The corresponding path in the a10-3588 development board is: /dev/spidev3.0

Driver code corresponding to spidev: kernel-5.10/drivers/spi/spidev.c

The kernel config needs to have the SPI\_SPIDEV selected:

```
| Symbol: SPI_SPIDEV [=y]
| Type : tristate
| Prompt: User mode SPI device driver support
| Location:
|   -> Device Drivers
|     -> SPI support (SPI [=y])
| Defined at drivers/spi/Kconfig:684
| Depends on: SPI [=y] && SPI_MASTER [=y]
```

DTS configuration is as follows:

```
&spi3 {
    pinctrl-names = "default";
    pinctrl-0 = <&spi3m0_cs0 &spi3m0_pins>;
    num-cs = <1>;
    status = "okay";

    spidev3: spidev@00{
        compatible = "rockchip,spidev";
        status = "okay";
        reg = <0x0>;
        spi-max-frequency = <50000000>;
    };
};
```

Please refer to kernel-5.10/Documentation/spi/spidev.rst for detailed instructions on using spidev.

## 6.14 UART

a) Device node

The left serial port is UART0, and the right serial port is UART1:

```
/dev/ttyWCH0
/dev/ttyWCH1
```

- b) Baud rate setting

Take UART0 as an example:

```
stty -F /dev/ttyWCH0 ispeed 115200 ospeed 115200 cs8
stty -F /dev/ttyWCH0 speed 115200 cs8 -parenb -cstopb -echo
```

- c) Send data

```
echo "aaa" > /dev/ttyWCH0
```

- d) Read the data

```
cat /dev/ttyWCH0
```

## 6.15 Watchdog Usage

This chapter mainly introduces the use of A10-3588 development board hardware watchdog.

### 6.15.1 DTS Configuration

The DTS node of watchdog for A10-3588 is defined in the kernel-5.10/arch/arm64/boot/dts/rockchip/rk3588-namts0-a10-3588.dtsi file as follows:

```
namts0_wdt {
    compatible = "linux,wdt-namts0";
    status = "okay";
    hw_margin_ms = <500>;
    hw-gpios = <&gpio0 RK_PB0 GPIO_ACTIVE_HIGH>;
    pinctrl-names = "default";
    pinctrl-0 = <&wdt_gpio>;
};
```

The watchdog is on by default, and the kernel automatically feeds the dog every 0.5 seconds.

### 6.15.2 Usage

The driver file for watchdog is kernel-5.10/drivers/watchdog/namts0\_wdt.c.

# Turn on the watchdog:

```
echo 1 > /sys/class/namts0_watchdog/enble
```

# Turn off the watchdog

```
echo 0 > /sys/class/namts0_watchdog/enble
```