Porting Guide (QuickRearCam)

Version 0.6.0

Display Audio

Solution Team

Release information

The following changes have been make to this document.

Change History

Date	Change
04 Dec. 2017	First release for v0.6.0

Proprietary Notice

Information in this document is provided solely to enable system and software implementers to use Nexell products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Nexell reserves the right to make changes without further notice to any products herein.

Nexell makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Nexell assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Nexell data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Nexell does not convey any license under its patent rights nor the rights of others. Nexell products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Nexell product could create a situation where personal injury or death may occur. Should Buyer purchase or use Nexell products for any such unintended or unauthorized application, Buyer shall indemnify and hold Nexell and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Nexell was negligent regarding the design or manufacture of the part.

Contact us

[11595] BundangYemiji Bldg. 12F, 31 Hwangsaeul-ro 258 beon gil, Bundang-gu, Sungnam-city, Gyeonggi-do, Korea.

TEL: 82-31-698-7400

FAX:82-31-698-7455

http://www.nexell.co.kr

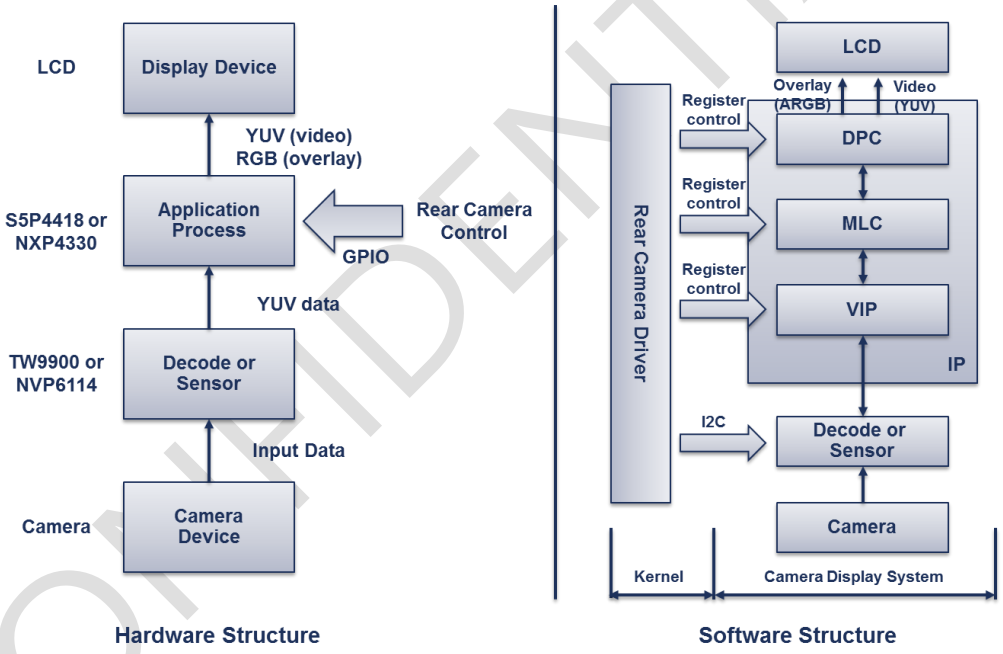
Contents

Chap 1. Implementation of Rear camera

1.1 Rear camera structure

Rear camera can output the image received via the external decoder (TW9900) or sensor (AHD) to the LCD according to the state of the GPIO (high or low) that controls whether the camera is operating or not. .

It also provides a Vendor API that can be processed at each point of operation and an API that can draw a parking extension line



1.2 Rear camera Vendor API

```
[Reference File]
[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam-vendor.h
[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam-vendor.c
```

1.2.1 nx_rearcam_alloc_vendor_context()

```
struct nx_vendor_context *nx_rearcam_alloc_vendor_context(void)
```

Description

Provides context for initialization at probe time.

Example

```
/* User data declaration */
struct nx_vendor_context {
    int type;
};

struct nx_vendor_context *nx_rearcam_alloc_vendor_context(void)
{
    /* User data type declaration to use */
    struct nx_vendor_context *ctx;
    ctx = kmalloc(sizeof(struct nx_vendor_context), GFP_KERNEL);
    if (!ctx)
        return NULL;
    /* Initialize context declared in user data */
    ctx->type = 1;
    return ctx;
}
```

1.2.2 nx_rearcam_free_vendor_context()

`void *nx_rearcam_free_vendor_context(void *)`
Description
It is an API that is provided to release resources used in Driver remove.

1.2.3 nx_rearcam_pre_turn_on()

`bool nx_rearcam_pre_turn_on(void *)`
Description
It is an API that provides the necessary work before the Rear camera operation. The return value is true and false. When returning false, the Rear camera does not work.

1.2.4 nx_rearcam_post_turn_off()

`void nx_rearcam_post_turn_off(void *)`
Description
It is an API that provides necessary work after Rear camera is stopped.

1.2.5 nx_rearcam_decide()

`bool nx_rearcam_decide(void *)`
Description
It is an API defined to define the processing to be performed when the GPIO signal that controls the motion of the Rear camera is generated.

1.2.6 nx_rearcam_sensor_init_func()

```
void nx_rearcam_sensor_init_func(struct i2c_client *client)
```

Description

It is a function to initialize the Senor that processes the work required to initialize the sensor to operate the Rear camera.

1.2.7 nx_rearcam_draw_rgb_overlay()

```
void nx_rearcam_draw_rgb_overlay(
        int width,
        int height,
        int pixelbyte,
        int rotation,
        void *ctx,
        void *mem
)
```

Description

It is an API that provides the parking guide line of Rear camera to be drawn. The parking guide line can be displayed by inputting the data of the parking guide line to the variable passing through the parameter.

Parameter

- width : width of display resolution
- height : height of display resolution
- pixelbyte : Pixelbyte in RGB format (ex: ARGB : pixelbyte = 4)
- rotation : Whether the screen is rotated.
- ctx : the vendor context that return in nx_rearcam_alloc_vendor_context function
- mem : Buffer to draw parking guides.

1.3 Rear camera Sensor initialization

```
[Reference File]
[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam.c
[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam-vendor.h
[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam-vendor.c
[kernel-4.4.x]/arch/arm/boot/dts/tw9900_rearcam_regset.dtsi
[kernel-4.4.x]/arch/arm/boot/dts/s5p4418-navi_ref-common.dtsi
```

Rear camera sensor initialization can be done by variables that set the register value and functions that can call initialization related functions.

If necessary, both methods can be initialized at the same time.

- Initialization with register initialization variable

Registered as “<address, Value>” to be initialized in the file, and initialized at the time of probe.

```
ex)
tw9900 decode initialization

Refer to tw9900_rearcam_regset.dtsi file
sensor_tw9900:register {
         data =  <0x02 0x40>,
         <0x1c 0x00>,
              .
              .
         <0xb1 0x20>;
};
```

- Initialization using initialization functions

It is necessary to register the function needed for sensor initialization as EXPORT_SYMBOL in Sensor driver so that it can be called.

It is called at the time of the device driver’s probe and is called immediately after the above “initialization using register initialization variable” occurs.

```
ex)
nvp6114a AHD Sensor initialization reference file

In the kernel-4.4.x / drivers / media / i2c / nvp / nvp6114a.c file
int nvp6114a_initialization(struct i2c_client *client)
{
……
}
EXPORT_SYMBOL(nvp6114a_initialization);

In the nx_rearcam_sensor_init_func function in the nx-rearcam-vendor.c file
Refer to the following.
void nx_rearcam_sensor_init_func(struct i2c_client *client)
{
#if defined(CONFIG_VIDEO_NVP6114A)
        extern void nvp6114a_initialization(struct i2c_client  *client);
        nvp6114a_initialization(client);
#endif
}
```

1.4 Draw parking guide line

[Reference file]

[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam.c

[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam-vendor.h

[kernel-4.4.x]/drivers/media/platform/nexell/capture/nx-rearcam-vendor.c

[kernel-4.4.x]/drivers/media/platform/nexell/capture/parking_line.h

Rear camera Use ARGB raw data to draw a parking line. This raw data provides a program that can be automatically created in an ARGB bmp file and can be created by this program.

For information on how to use this program, refer to “Rear camera parking guide line application document”.

The generated parking guide line files are generated by the program, and the width and height of each file are created with an array variable named “parking line resolution” in the order of creation and registration.

The raw data is generated with an array variable named “parkingline_filename” .

Depending on the user’s preference, this header file can be assigned to the last parameter buffer (mem) in the nx_rearcam_draw_rgb_overlay function, which is an API to draw parking guide lines provided in the file nx-rearcam-vendor.c.

```
ex)
The parking_line.h file created with the ARGB 32bit bmp files
(left, bmp, center.bmp, right.bmp)


*. The contents of the parking_line.h file


const unsigned int parkingline_resolution[3][2] = {
    { 1024, 600 },
    { 1024, 600 },
    { 1024, 600 }
};


const unsigned int parkingline_left[] = {
    ……
};
const unsigned int parkingline_center[] = {
    ……
};const unsigned int parkingline_right[] = {
    ……
};
```

```
*. Draw a parking guide line in the nx_rearcam_draw_rgb_overlay function
   to draw a parking line
void nx_rearcam_draw_rgb_overlay(int width, int height, int pixelbyte,
                                 int rotation, void *ctx, void *mem)
{
    if (mem != NULL) {
        int i, j;
        int src_width, src_height;
        static int line_index;
        unsigned int *data;

        /* Buffer to draw parking guide line */
        u32 *pbuffer = (u32 *)mem;

        line_index = line_index % 3;

        /* Get the width and height of the generated raw file.  */
        src_width = parkingline_resolution[line_index][0];
        src_height = parkingline_resolution[line_index][1];

        switch(line_index) {
        case 0:
            /* Parking guide line generated from left.bmp raw data */
            data = parkingline_left;
            break;
        case 1:
            /* Parking guide line generated from center.bmp raw data  */
            data = parkingline_center;
            break;
        case 2:
            /* Parking guide line generated from right.bmp raw data */
            data = parkingline_right;
            break;
        }

        memset(mem, 0, width * height * pixelbyte);

        {
            for (i = 0; i < src_height; i++)
                for (j = 0; j < src_width; j++) {
                    /* Draw parking guide line raw data in buffer. */
                    pbuffer[i * width + j] =
```

```
                        data[i * src_width + j];
                }
            }


            line_index++;
    }
}
```

1.5 Rear camera configuration

```
[Reference file]
[kernel-4.4.x]/arch/arm/boot/dts/s5p4418-navi_ref-common.dtsi
[kernel-4.4.x]/arch/arm/boot/dts/s5p4418-pinctrl.dtsi
[kernel-4.4.x]/arch/arm/boot/dts/s5p4418.dtsi
```

Rear camera requires various settings for operation. Here, the meanings of the set values and the set values will be described.

To enable the rear camera driver included in the kernel, set the following in the file “s5p4418-navi_ref-common.dtsi”.

The following describes the settings and meanings of the device tree set in the file “s5p4418-navi_ref-common.dtsi”.

1.5.1 Setting Rear camera using tw9900 decode

```
#include “tw9900_rearcam_regset.dtsi”
&rearcam {
 /*
      Decode h / w interface pin (sync and data) setting.
      Vid2_data_clk is the pin control device tree node set in the s5p4418-pinctrl.dtsi
file,
      and the configuration can be checked in s5p4418-pinctrl.dtsi.
*/
       pinctrl-names = “default”;
       pinctrl-0 = <&vid2_data_clk>;


 /*
     Camera rotation – 0: do not rotate, 1: rotate
*/
       rotation = <0>; /*
      VIP clipper information of the camera to be used
      in the rear camera among the registered cameras is fetched.
*/
       rear_cam_dev = <&clipper_1>;
```

```
/*
      Tw9900 sensor setting value.
      Declared in tw9900_rearcam_regset.dtsi.
*/
        sensor_reg = <&sensor_tw9900>;


/*
      Reads the set display (dpc, mlc) setting value.
      The dp_drm node is defined in s5p4418.dtsi and s5p4418-navi_ref-common.dtsi.
*/
        display = <&dp_drm>;



/*
       Reads the set display device (lcd) setting value.
      The dpdrm node is defined in s5p4418.dtsi and s5p4418-navi_ref-common.dtsi.
*/
        dp_drm_dev = <&dp_drm_lvds>;



/*
      rear camera video width
*/
        width = <704>;

/*
      rear camera video height
*/
        height = <480>;

/*
      enable rear camera.
*/
        status = “okay”;

/*
      Rear camera GPIO setting to operate
*/
```

```
        gpio {
                /* event GPIO setting  */
                event-gpio = <&alive_0 3 0>;
                /* active polar setting, high active : 1, low active :0  */
                active_high = <0>;
              /* Delay from gpio event to rear camera operation  */
                detect_delay = <0>;
        };


/*
      rear camera MLC(Multi Layer Controller) setting
*/
        mlc {
                /* Set the module number of MLC Primary: 0, Second: 1  */
                module = <0>;

                /*
               RGB format (ARGB) used to draw parking guide lines.
               - Supported formats
                     nx_mlc_rgbfmt_a1r5g5b5 = 0x33420000ul,
                     nx_mlc_rgbfmt_a1b5g5r5 = 0xb3420000ul,
                     nx_mlc_rgbfmt_a4r4g4b4 = 0x22110000ul,
                     nx_mlc_rgbfmt_a4b4g4r4 = 0xa2110000ul,
                     nx_mlc_rgbfmt_a8r3g3b2 = 0x11200000ul,
                     nx_mlc_rgbfmt_a8b3g3r2 = 0x91200000ul,
                     nx_mlc_rgbfmt_a8r8g8b8 = 0x06530000ul,
                     nx_mlc_rgbfmt_a8b8g8r8 = 0x86530000ul
               */
                format = <0x06530000>;
        };

 /*
      rear camera DPC(DisPlay Controller) setting
*/
        dpc {
                 /* DPC module number setting Primary: 0, Second: 1 */
                module = <0>;
        };
```

1.5.2 Configuring VIP Clipper with tw9900 decode

```
&clipper_1 {
 /*
       camera sensor interface type
      - MIPI : NX_CAPTURE_INTERFACE_MIPI
      - Parallel : NX_CAPTURE_INTERFACE_PARALLEL
                          parallel interface is 8bit(656/601)
*/
        interface_type = <NX_CAPTURE_INTERFACE_PARALLEL>;



/*
       Decode h / w interface pin (sync and data) setting.
       Vid2_data_clk is the pin control device tree node set in the s5p4418-pinctrl.dtsi
file,
       and the configuration can be checked in s5p4418-pinctrl.dtsi.
*/
        pinctrl-names = “default”;
        pinctrl-0 = <&vid2_data_clk>;


/*
      Declaring the GPIO to use to initialize the Sensor
*/
        gpios = <&gpio_c 9 0 &gpio_e 16 0 &gpio_e 12 0>;


/*
      The data order from the Sensor
       - Supported formats
              NX_VIN_CBY0CRY1,
              NX_VIN_CRY1CBY0,
              NX_VIN_CRY1CBY0,
              NX_VIN_Y1CRY0CB
*/
        data_order = <NX_VIN_CBY0CRY1>;


/*
      Video Input Processor (VIP) Input port: 0 or 1
*/
        port = <1>;
/*
      Set whether input interface is 601 or 656 – 1: 601, 0: 656
*/
```

```
        external_sync = <0>;



/*
    Set whether input data is interlace or progressive. 1: interlace, 0: progressive
*/
        interlace = <1>;



/*
     enable device driver
*/
        status = “okay”;

        sensor {
                /* Senor type: Fixed as NX_CAPTURE_SENSOR_I2C */
                type = <NX_CAPTURE_SENSOR_I2C>;
                /* sensor name  */
                i2c_name = “tw9900”;
                /* sensor i2c adapter number  */
                i2c_adapter = <5>;
                /* sensor i2c address  */
                addr = <0x44>;
        };

        power {

 /*
     Camera sensor power Turn on control sequence
     *. [ACTION_START_TAG][ACTION_TYPE][SEQUENCE][ACTION_END_TAG]
         - ACTION_START_TAG : NX_ACTION_START
         - ACTION_END_TAG : NX_ACTION_END
         - ACTION_TYPE : NX_ACTION_TYPE_PMIC, NX_ACTION_TYPE_GPIO, NX_ACTION_TYPE_CLOCK
               ACTION_TYPE_GPIO control sequence : gpionum, value, delay_ms, value,
delay_ms …
               ACTION_TYPE_PMIC control sequence : regulator_name, value, delay_ms
               ACTION_TYPE_CLOCK control sequence : enable/disable, delay_ms
*/
              enable_seq = <
              NX_ACTION_START NX_ACTION_TYPE_GPIO 2 0 1 0 10 NX_ACTION_END
              NX_ACTION_START NX_ACTION_TYPE_GPIO 1 0 1 0 10 NX_ACTION_END
```

```
                NX_ACTION_START NX_ACTION_TYPE_GPIO 0 1 10 NX_ACTION_END
                >;



/*
      Camera sensor power Control sequence when turning off
      Control is the same as the description “control sequence
       when camera sensor power is turned on”.
*/
      disable_seq = <
            NX_ACTION_START NX_ACTION_TYPE_GPIO 2 0 1 0 10 NX_ACTION_END
            NX_ACTION_START NX_ACTION_TYPE_GPIO 1 0 1 0 10 NX_ACTION_END
            NX_ACTION_START NX_ACTION_TYPE_GPIO 0 1 10 NX_ACTION_END
            >;
        };

      clock {
                        /* camera sensor master clock pwm setting */
                        pwms = <&pwm_device1 0 24000000 0>
          };
};
```

1.5.3 LVDS Display device setting

```
&dp_drm_lvds {
        status = “ok”;          display-timing {
                 /* input clock frequency */
                clock-frequency = <50000000>;
                 /* input data width */
                hactive = <1024>;
                 /* input data height */
                vactive = <600>;
                 /* input data horizontal front porch */
                hfront-porch = <160>;
                 /* input data horizontal back porch */
                hback-porch = <140>;
                 /* input data horizontal sync */
                hsync-len = <20>;
                 /* input data vertical front porch */
                vfront-porch = <12>;
                 /* input data vertical back porch */
                vback-porch = <20>;
```

```
                /* input data vertical sync */
                vsync-len = <3>;
        };

        dp_control {
 /*
       Clock generator ‘0’ display clock source setting
      – 0:PLL0, 1:PLL1, 2:SVLCK, 3:P(S)VCLK, 4:~P(S)VCLK, 5:AVCLK, 6:~SVLCK, 7:ClKGEN0
*/
                clk_src_lv0 = <0>;
 /*
       Clock generator ‘0’ display clock divisor setting (1 to 256)
*/
                clk_div_lv0 = <16>;
 /*
       Display clock source setting of clock generator ‘1’
      – 0:PLL0, 1:PLL1, 2:SVLCK, 3:P(S)VCLK, 4:~P(S)VCLK, 5:AVCLK, 6:~SVLCK, 7:ClKGEN0
*/
                clk_src_lv1 = <7>;
/*
       Clock generator ‘1’ display clock divisor setting (1 to 256)
*/
                clk_div_lv1 = <1>;

/*
       display out format setting
*/
                out_format = <2>;
        };
};
```

Chap 2. Porting of Rear camera

2.1 Copy and modify files

copy or modify files related to Rear camera shown in the table below.

File	Path	Description
nx-rearcam.c	[kernel-4.4.x]/drivers/media/platform/nexell/capture	Rear camera driver implementation file
nx-rearcam-vendor.c	[kernel-4.4.x]/drivers/media/platform/nexell/capture	Rear camera vedor API implementation file
nx-rearcam-vendor.h	[kernel-4.4.x]/drivers/media/platform/nexell/capture	Rear camera vedor API header file
parking_line.h	[kernel-4.4.x]/drivers/media/platform/nexell/capture	Parking guide line data declaration header file
tw9900_rearcam_regset.dtsi	[kernel-4.4.x]/arch/arm/boot/dts	tw9900 sensor data definition file

2.2 Edit Make File

Adds the line below to kernel-4.4.x/drivers/media/platform/nexell/capture/Make file

```
[Additional line]
obj-$(CONFIG_VIDEO_NEXELL_REARCAM)      += nx-rearcam-vendor.o nx-rearcam.o
```

2.3 Modify Kconfig File

Adds the line below to kernel-4.4.x/drivers/media/platform/nexell/capture/Kconfig file

```
[Additional line]
config VIDEO_NEXELL_REARCAM
bool “Enable fast preview of rear camera device”
default n
—help—
Say y here, you cna view cmaera as soon as kernel booting, and this
        driver switch on/off camera by gpio interrupt.
```

2.4 Menuconfig registration

2.4.1 enable Rear Camera drive

```
~/Kernel4.4 $ make ARCH=arm menuconfig
```

```
[location]
```

```
-> Device Drivers
            -> Multimedia support (MEDIA_SUPPORT
                        -> V4L platform devices
                                    -> Nexell S5Pxx18 SoC series V4l2 Subsystem driver
                                                -> Nexell S5Pxx18 SoC series camera
interface driver
                                                            [*] Enable fast preview of
rear camera device
```

2.4.2 Copy the config file

```
~/kernel-4.4.x $ cp .config arch/arm/configs/s5p4418_navi_ref_defconfig
```

2.5 Rear camera configuration setting

Add the following to kernel-4.4.x / arch / arm / boot / dts / s5p4418-navi_ref-common.dtsi file

```
1. tw9900 decode configuration setting
&clipper_1 {
    interface_type = <NX_CAPTURE_INTERFACE_PARALLEL>;
    pinctrl-names = “default”;
    pinctrl-0 = <&vid2_data_clk>;
    gpios = <&gpio_c 9 0 &gpio_e 16 0 &gpio_e 12 0>;
    data_order = <NX_VIN_CBY0CRY1>;
    port = <1>;
    external_sync = <0>;
    interlace = <1>;
    status = “okay”;    sensor {
        type = <NX_CAPTURE_SENSOR_I2C>;
        i2c_name = “tw9900”;
        i2c_adapter = <5>;
        addr = <0x44>;
    };    power {
        enable_seq = <
        NX_ACTION_START NX_ACTION_TYPE_GPIO 2 0 1 0 10 NX_ACTION_END
        NX_ACTION_START NX_ACTION_TYPE_GPIO 1 0 1 0 10 NX_ACTION_END
        NX_ACTION_START NX_ACTION_TYPE_GPIO 0 1 10 NX_ACTION_END
        >;
    };
};

#include “tw9900_rearcam_regset.dtsi”
&rearcam {
```

```
    pinctrl-names = “default”;
    pinctrl-0 = <&vid2_data_clk>;
    rotation = <0>;
    rear_cam_dev = <&clipper_1>;
    sensor_reg = <&sensor_tw9900>;
    display = <&dp_drm>;
    dp_drm_dev = <&dp_drm_lvds>;
    width = <704>;
    height = <480>;
    status = “okay”;    gpio {
        event-gpio = <&alive_0 3 0>;
        active_high = <0>;
        detect_delay = <0>;
    };    mlc {
        module = <0>;
        format = <0x06530000>;
    };    dpc {
        module = <0>;
    };
};
```

2. nvp6114a AHD sensor configuration setting

```
&clipper_1 {
    interface_type = <NX_CAPTURE_INTERFACE_PARALLEL>;
    pinctrl-names = “default”;
    pinctrl-0 = <&vid2_data_clk>;
    gpios = <&gpio_b 12 0 &gpio_c 10 0 &gpio_c 9 0 &gpio_b 14 0>;
    data_order = <NX_VIN_CBY0CRY1>;
    port = <1>;
    external_sync = <0>;
    interlace = <0>;
    status = “okay”;

    sensor {
        type = <NX_CAPTURE_SENSOR_I2C>;
        i2c_name = “nvp6114a”;
        i2c_adapter = <3>;
        addr = <0x32>;
    };
```

```
    power {
        enable_seq = <
        NX_ACTION_START NX_ACTION_TYPE_GPIO 3 1 10 NX_ACTION_END
        NX_ACTION_START NX_ACTION_TYPE_GPIO 2 0 10 NX_ACTION_END
        NX_ACTION_START NX_ACTION_TYPE_GPIO 1 1 10 NX_ACTION_END
        NX_ACTION_START NX_ACTION_TYPE_GPIO 0 1 10 NX_ACTION_END
        >;
    };
};

&rearcam {
    rotation = <0>;
    skip_frame = <5>;
    pinctrl-names = “default”;
    pinctrl-0 = <&vid2_data_clk>;
    rear_cam_dev = <&clipper_1>;
    display = <&dp_drm>;
    dp_drm_dev = <&dp_drm_lvds>;
    width = <1280>;
    height = <720>;
    status = “okay”;

    gpio {
        event-gpio = <&alive_0 3 0>;
        active_high = <0>;
        detect_delay = <0>;
    };

    mlc {
        module = <0>;
        format = <0x06530000>;
    };

    dpc {
        module = <0>;
    };
};
```


2.6 Rear camera compile

Compile with the following command.

```
~/D-Audio/yocto-20170418$ ./tools/build.sh s5p4418-navi-ref qt -t kernel
```