Camera Bring Up User Guide

Revision: 0.1

Release Date: 2022-05-07

Contact Information

- Website: www.amlogic.com
- Pre-sales consultation: contact@amlogic.com
- Technical support: support@amlogic.com

Reader

This document is intended for the following readers:

- Field Application Engineer
- Software Development Engineer

Revision Record

Version	Comment
0.1 (2022-05-07)	1st release

Abbreviations

isp	image signal processing
sdk	software development kit
dts	device tree source
iq	image quality
v4l2	video for linux2
fr	full resolution
ds1	down scale channel 1
ds2	down scale channel 2
wdr	wide dynamic range

Directory

1 Overview

This document describes how to bring up one Camera Module. That introduces the features to be updated & added, and the verification method.

2 Camera Bring Up

2.1 isp

- **isp source directory structure**

```
buildroot code path: hardware/aml-x.xx/amlogic/arm_isp/isp_module/
android code path: /vendor/amlogic/common/arm_isp/driver/linux/kernel/
There are one Makefile file and two sub floders "subdev/" "v4l2_dev/" within
the above path.
Three drivers ("iq", "sensor", "lens") are in "subdev" path,
And "v4l2 Framework" is in "v4l2_dev" path.
```

To bring up one module only needs to configure at the **sensor** directory and **iq** directory. The purpose for **sensor** directory is to bring up camera, and the purpose for **iq** directory is to debug the iq of sensor.

- **Compiling method for isp code**

```
Enter into isp code directory, and change the compiling tool path of
Makefile
Execute "make": all 4 modules will be compiled;
Execute "make clean": all files which generated by 4 modules will be erased;

Only compile one module: make v4l2;  make iq;  make sensor; make lens
Only erase one module: make v4l2_clean make iq_clean;  make sensor;  make
lens_clean
```

Notice: place generated 4 ko files into "/vendor/lib/modules" for Android devices. if it is Linux device, please place ko file into "/lib/modules".

2.2 dts config

SDK DTS file for buildroot is in "kernel/aml-x.xx/arch/armx/boot/dts/amlogic/"

SDK DTS file for Android is in "common/arch/armx/boot/dts/amlogic/"

- **sensor node**

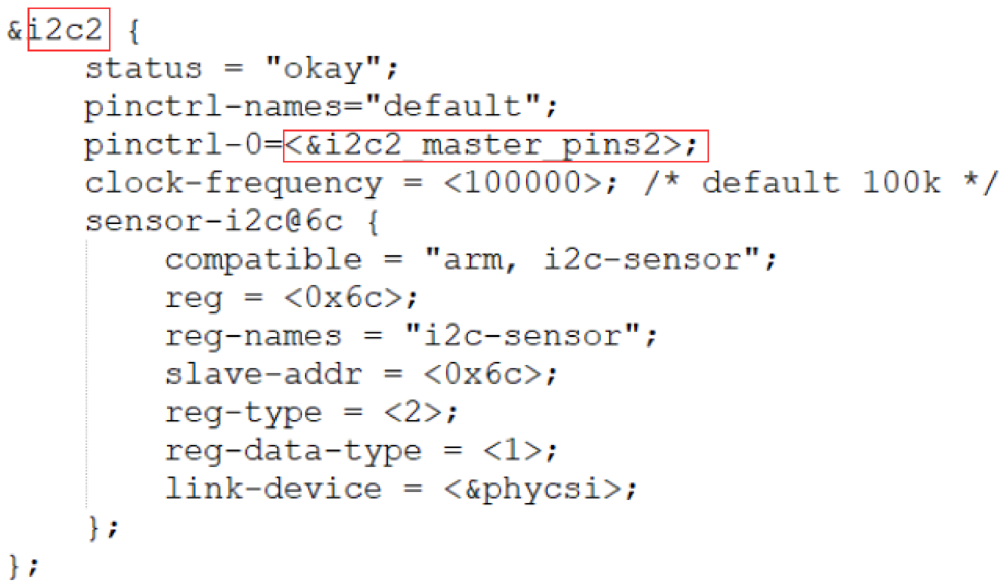
In **dts** file, those need to be updated under **sensor** node:

sensor-name: update by actual sensor type, which should be aligned with the xxx_drv's sensor-name mentioned in later chapter.

clk: the clk value from SDK is 24MHz as default. So it is suggested that sensor clock setting can be configured as 24MHz by Module factory.

reset: the **reset** feature for sensor. Please update the reset pin based on the hardware configuration of the board.

- **i2c node**



```
&i2c2 {
    status = "okay";
    pinctrl-names="default";
    pinctrl-0=<&i2c2_master_pins2>;
    clock-frequency = <100000>; /* default 100k */
    sensor-i2c@6c {
        compatible = "arm, i2c-sensor";
        reg = <0x6c>;
        reg-names = "i2c-sensor";
        slave-addr = <0x6c>;
        reg-type = <2>;
        reg-data-type = <1>;
        link-device = <&phycsi>;
    };
};
```

As shown above, i2c2 is used as default, i2c2_master_pins2 is set as pinctrl. If customer do use i2c, rather than i2c2, please set i2c in above red cycle content. 0x6c represents the i2c address of sensor, and the value (0x6c) can not be updated, for isp driver will read the address from each sensor's ***_config.h file.

- **adapter node**

```
 /* Calculation Way */
 adapter memory size = fr width * fr height * sensor depth / 8

 /* Calculation Description */
 Take the sensor for 3840*2160 as example,
 fr width = 3840, fr height = 2160, sensor depth = 12 /* Each sensor's
sensor depth can be estimated as 12bit for calculation */
 adapter memory size = 3840 * 2160 * 12 / 8 = 12M /* adapter memory size
need to be aligned as 4M */
 /* In adapter, the size of mem_alloc and adapt_cma_reserved need to be set
as 12.The setting should be aligned consistently */

 /* Configuration Example */
 /* 1920*1080 */
 &adapter {
     status = "okay";
     mem_alloc = <4>;
     memory-region = <&adapt_cma_reserved>;
 };
 adapt_cma_reserved:linux,adapt_cma {
     compatible = "shared-dma-pool";
     status = "okay";
     size = <0x0 0x0400000>;
 };

 /* 3840*2160 */
 &adapter {
     status = "okay";
     mem_alloc = <12>;
     memory-region = <&adapt_cma_reserved>;
 };
```

```
    adapt_cma_reserved:linux,adapt_cma {
        compatible = "shared-dma-pool";
        status = "okay";
        size = <0x0 0x0C00000>;
        };
    };
```

- **isp node**

```
    /* Calculation Way */
    temper-line-offset =  fr width * 3（Unit:bit）
    temper-buf-size =  fr width * fr height * 6（Unit:M）
    temper-frame-num = <1>, temper 2 == 1, temper 3  == 2
    temper-frame-size = fr width * fr height * 6（Unit:bit）

    /* Calculation Description */
    /* 3840*2160 */
    temper-line-offset =  3840 * 3 = 11520 = 0x2D00
    temper-buf-size =  3840 * 2160 * 6 = 48M
    temper-frame-num = <1> /* Always set as 1 */
    temper-frame-size = 3840 * 2160 * 6 = 48M = 0x3000000

    /* Configuration Example */
    /* 1920*1080 */
    &isp {
        status = "okay";
        temper-line-offset = <0x1680>;
        temper-buf-size = <12>;
        temper-frame-num = <1>;
        temper-frame-size = <0x0c00000>;
        memory-region = <&isp_cma_reserved>;
    };

    /* 3840*2160 */
    &isp {
        status = "okay";
        temper-line-offset = <0x2D00>;
        temper-buf-size = <48>;
        temper-frame-num = <1>;
        temper-frame-size = <0x3000000>;
        memory-region = <&isp_cma_reserved>;
    };
```

- **isp_cma node**

```
    /* Calculation Way */
    fr size = fr width * fr height * fr fourcc * 6
    ds1 size = ds1 width * ds1 height * ds1 fourcc * 6
    ds2 size = ds2 width * ds2 height * ds2 fourcc * 6
    temper-buf-size =  fr width * fr height * 6
    Total size = fr size + ds1 size + ds2 size + temper-buf-size

    /* Calculation Description */
    /*
    fourcc is the output of image format. When the output is nv21, fourcc
  equals to 1.5，while the output is RGB24, fourcc equals to 3.
  fr: 3840 * 2160, ds1: 1920 * 1080
  Currently, "ds1 1080p nv21" is used for preview, "fr 3840*2160 RGB24" is
used for taking photo. It is suggested to reserve additionl 16M memory for
total size.
  */
  fr size = 3840 * 2160 * 3 * 6 = 144M
  ds1 size = 1920 * 1080 * 1.5 * 6 = 18M
  ds2 size = 0
  temper-buf-size =  3840 * 2160  * 6 = 48M
  Total size = 144+ 18 + 48 = 212M (Need 4M for alignment)

  /* Configuration Example */
  /* 1920*1080 */
  isp_cma_reserved:linux,isp_cma {
      compatible = "shared-dma-pool";
      status = "okay";
      size = <0x5400000>;
      alignment = <0x400000>;
  };

  /* 3840*2160 */
  isp_cma_reserved:linux,isp_cma {
      compatible = "shared-dma-pool";
      status = "okay";
      size = <0xd800000>;
      alignment = <0x400000>;
  };
```

2.3 driver config

- **iq driver**

  Please change the below file when a new sensor is needed to be adapted,

```
  subdev/iq/app/soc_iq.c
  subdev/iq/src/calibration/acamera_calibrations_static_linear_xxx.c
  subdev/iq/src/calibration/acamera_calibrations_dynamic_linear_xxx.c
  subdev/iq/src/calibration/acamera_calibrations_static_fs_lin_xxx.c
  subdev/iq/src/calibration/acamera_calibrations_dynamic_fs_lin_xxx.c
  subdev/iq/src/calibration/acamera_get_calibrations_dummy.c
  subdev/iq/src/calibration/acamera_get_calibrations_dummy.h
  subdev/iq/src/fw_runtime_initialization_settings.h
```

acamera_calibrations_static_linear_xxx.c has the objective parameters which are generated from those raw pictures taken by Calibration Tool, and belong to the parameters in linear mode.
acamera_calibrations_dynamic_linear_xxx.c has the parameters which are generated in online debug process by Control Tool, and belong to the parameters in linear mode.
acamera_calibrations_static_fs_lin_xxx.c has the objective parameters which are generated from those raw pictures taken by Calibration Tool, and belong to the parameters in wdr mode.
acamera_calibrations_dynamic_fs_lin_xxx.c has the parameters which are generated in online debug process by Control Tool, and belong to the parameters in wdr mode.

During the Bring Up phase, it only needs to do copy and rename these 4 configured and same type of sensors' effective parameter files.

If sensor doesn't support wdr mode, it doesn't need to migrate the 2 effective files with "fs_lin".

In **soc_iq.c**, it needs to add a group of relevant sensor in iqConversionTable array.

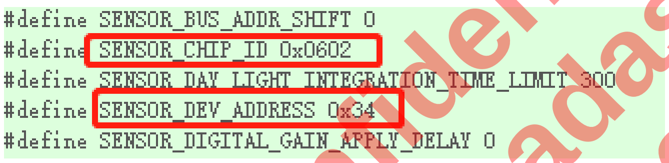
In **acamera_get_calibrations_dummy.c**, it needs to define the get_calibrations function of the relevant sensor, please refer to the existed module file for adaption.
In **acamera_get_calibrations_dummy.h**, it needs to make the external claim for the above 4 parameter files.
In **runtime_initialization_settings.h**, it needs to make the external claim for get_calibrations function defined in acamera_get_calibrations_dummy.c

- **sensor driver**

To adapt new sensor with iq driver, below files should be updated:

```
    subdev/sensor/app/soc_sensor.c
    subdev/sensor/src/driver/sensor/xxx_config.h
    subdev/sensor/src/driver/sensor/xxx_drv.c
    subdev/sensor/src/driver/sensor/xxx_seq.h
subdev/sensor/src/fw/runtime_initialization_settings.h
```

xxx_config.h is used to define some Macro for driver. Mainly, the sensor's "chip_id" and "i2c address" values should be changed. These 2 values can be checked from the sensor's datasheet. The i2c address of platform is 8bit, as shown below:



```
#define SENSOR_BUS_ADDR_SHIFT 0
#define SENSOR_CHIP_ID 0x0602
#define SENSOR_DAY_LIGHT_INTEGRATION_TIME_LIMIT 300
#define SENSOR_DEV_ADDRESS 0x34
#define SENSOR_DIGITAL_GAIN_APPLY_DELAY 0
```

xxx_seq.h is the sensor's register configuration, which need to retrieve from sensor vendor.

xxx_drv.c is sensor's driver file. Mainly, supported_modes[] need to be changed. And read/write register should be changed to the sensor's register, e.g. the register of vmax, hmax, intergration time, gain, stream on/off, chip id etc...,

```
/*
Update wdr_mode = WDR_MODE_LINEAR based on sensor's setting
Resolution: the resolution output bits set by Camera. As to the format bits
outputted by camera, the isp from raw8, raw 10, raw16, w400 platfrom usually
only support to receive RAW format data, but other format data.
exposures: 1:linear 2:wdr
lanes bps is configured based on setting value.
bayer datasheet has some instructment.
num is configured based on setting value.
*/

static sensor_mode_t supported_modes[] = {
    {
        .wdr_mode = WDR_MODE_LINEAR,
        .fps = 30 * 256,
        .resolution.width = 3840,
        .resolution.height = 2160,
        .bits = 10,
        .exposures = 1,
        .lanes = 4,
        .bps = 720,
        .bayer = BAYER_GBRG,
        .dol_type = DOL_NON,
        .num = 0,
    },
}
static void sensor_update( void *ctx )
{
    /* Simulation Register Setting is listed as below, it only needs to
change the register address. */
    acamera_sbus_write_u8( p_sbus, 0x3091, (reg_gain >> 8 ) & 0x07 );
    acamera_sbus_write_u8( p_sbus, 0x3090, (reg_gain >> 0 ) & 0xFF );
    /* Explosure Resgister Setting is listed as below. */
    /* Short explosure */
    acamera_sbus_write_u8( p_sbus, 0x3051, ( p_xxx->shr0_old >> 8 ) & 0xFF
);
    acamera_sbus_write_u8( p_sbus, 0x3050, ( p_xxx->shr0_old >> 0 ) & 0xFF
);
    /* Long explosure */
    acamera_sbus_write_u8( p_sbus, 0x3055, ( p_xxx->shr1_old >> 8 ) & 0xFF
);
  acamera_sbus_write_u8( p_sbus, 0x3054, ( p_xxx->shr1_old >> 0 ) & 0xFF );
}
```

xxx_config.h、**xxx_seq.h**、**xxx_drv.c** 3 drivers, which can copy and update the same type of sensor files which have been adapted in SDK.

In **soc_sensor.c**, current module needs to add one group of sensor into ConversionTable array.

In **runtime_initialization_settings.h**, it needs to make the external claim and relationship of the relevant "module's sensor_init", "sensor_deinit", and "sensor_detect".

3 dual sensor Bring up

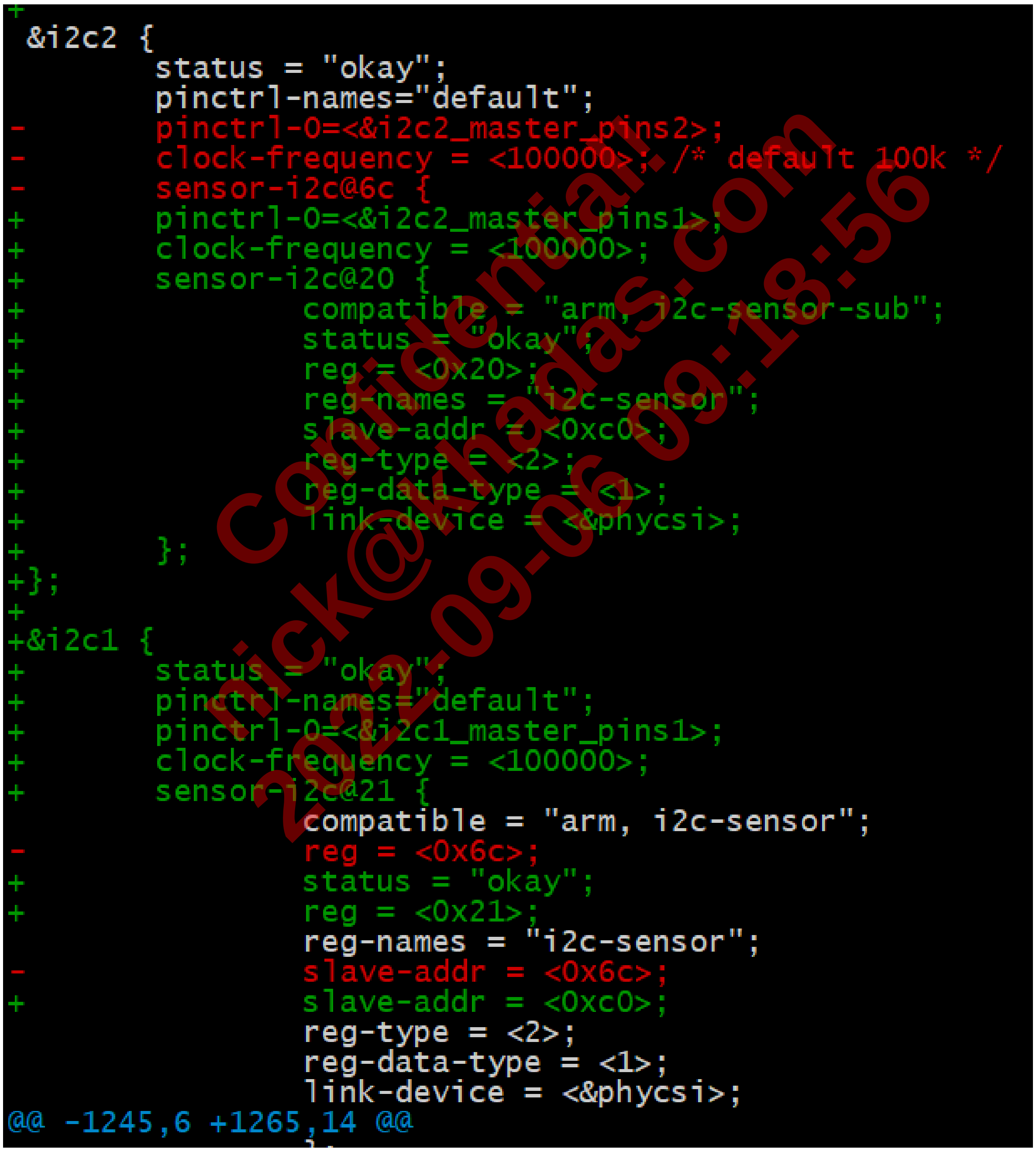
The code deployment structre, compile method, generated file of dual sensor is as same as the single sensor's.

3.1 dts config

Double size for the relevant DTS's isp adapter configuration.

3.2 Updating i2c node

Due to dual mipi sensor, i2c also becomes 2 pcs, please take care of i2c pin configuration.



```
+
 &i2c2 {
        status = "okay";
        pinctrl-names="default";
-       pinctrl-0=<&i2c2_master_pins2>;
-       clock-frequency = <100000>; /* default 100k */
-       sensor-i2c@6c {
+       pinctrl-0=<&i2c2_master_pins1>;
+       clock-frequency = <100000>;
+       sensor-i2c@20 {
+               compatible = "arm, i2c-sensor-sub";
+               status = "okay";
+               reg = <0x20>;
+               reg-names = "i2c-sensor";
+               slave-addr = <0xc0>;
+               reg-type = <2>;
+               reg-data-type = <1>;
+               link-device = <&phycsi>;
+       };
+};
+
+&i2c1 {
+       status = "okay";
+       pinctrl-names="default";
+       pinctrl-0=<&i2c1_master_pins1>;
+       clock-frequency = <100000>;
+       sensor-i2c@21 {
                compatible = "arm, i2c-sensor";
-               reg = <0x6c>;
+               status = "okay";
+               reg = <0x21>;
                reg-names = "i2c-sensor";
-               slave-addr = <0x6c>;
+               slave-addr = <0xc0>;
                reg-type = <2>;
                reg-data-type = <1>;
                link-device = <&phycsi>;
@@ -1245,6 +1265,14 @@
```

3.3 Updating iq driver

- **Files need to be updated**

The file update way for Iq file is same as the single sensor. It needs to add one additional group of xxx_sub's sensor driver when adding sensor name.

```
  uint32_t get_calibrations_xxx( uint32_t ctx_id, void *sensor_arg,
AcameraCalibrations *c )
  {
      uint8_t ret = 0;

      if ( !sensor_arg ) {
          LOG( LOG_CRIT, "calibration sensor_arg is NULL" );
          return ret;
      }

      int32_t preset = ( (sensor_mode_t *)sensor_arg )->wdr_mode;
      /* logic which calibration to apply */
      switch ( preset ) {
          case WDR_MODE_LINEAR:
              LOG( LOG_DEBUG, "calibration switching to WDR_MODE_LINEAR %d
", (int)preset );
              ret += ( get_calibrations_dynamic_linear_xxx( c ) +
get_calibrations_static_linear_xxx( c ) );
          break;
          case WDR_MODE_NATIVE:
              LOG( LOG_DEBUG, "calibration switching to WDR_MODE_NATIVE %d
", (int)preset );
          break;
          case WDR_MODE_FS_LIN:
              LOG( LOG_DEBUG, "calibration switching to WDR mode on mode %d
", (int)preset );
              ret += ( get_calibrations_dynamic_fs_lin_dummy( c ) +
get_calibrations_static_fs_lin_dummy( c ) );
          break;
          default:
              LOG( LOG_DEBUG, "calibration defaults to WDR_MODE_LINEAR %d ",
(int)preset );
              ret += ( get_calibrations_dynamic_linear_xxx( c ) +
get_calibrations_static_linear_xxx( c ) );
          break;
      }

      return ret;
  }

  uint32_t get_calibrations_xxx_sub( uint32_t ctx_id, void *sensor_arg,
AcameraCalibrations *c )
  {
      uint8_t ret = 0;
      if ( !sensor_arg ) {
          LOG( LOG_CRIT, "calibration sensor_arg is NULL" );
          return ret;
      }

      int32_t preset = ( (sensor_mode_t *)sensor_arg )->wdr_mode;
      /* logic which calibration to apply */
      switch ( preset ) {
      case WDR_MODE_LINEAR:
```

```
                LOG( LOG_DEBUG, "calibration switching to WDR_MODE_LINEAR %d
", (int)preset );
                ret += ( get_calibrations_dynamic_linear_xxx_sub( c ) +
get_calibrations_static_linear_xxx_sub( c ) );
            break;
            case WDR_MODE_NATIVE:
                LOG( LOG_DEBUG, "calibration switching to WDR_MODE_NATIVE %d
", (int)preset );
            break;
```

subdev/iq/src/fw/runtime_initialization_settings.h

Just same as the update way for single sensor, it needs to add one more group of xxx_sub function.

- **Files need to be added**

```
1、acamera_calibrations_dynamic_linear_xxx_sub.c
2、acamera_calibrations_static_linear_xxx_sub.c
3、acamera_calibrations_static_linear_xxx.c
4、acamera_calibrations_dynamic_linear_xxx.c
```

These files may be copied from other files and updated for iq debug later. Above 4 files are all for linear mode, if wdr mode is needed, please add more 4 files.

3.4 Updating sensor driver

- **Files need to be updated**

  Just same as the update way for single sensor, it needs to add one more group of xxx_sub's sensor.

- **Files need to be added**

  Directly copy the files from provided SDK and configure based on the relevant sensor situation. The register configuration is same as the single sensor configuration.

```
subdev\sensor\src\driver\sensor\xxx_config.h
subdev\sensor\src\driver\sensor\xxx_drv.c
subdev\sensor\src\driver\sensor\xxxSub_drv.c
subdev\sensor\src\driver\sensor\xxx_seq.h
```

4 sensor debug

4.1 Camera Can't be Brought Up

- Check whether driver matches with dts, and confirm that driver has called **probe**.
- If driver has called **probe**, please check whether these 3 hardware power items "avdd, dvdd, and iovdd" are supplied by normal hardware power or software control power. If they are supplied by software control power, it needs to add the control pin to these 3 power items.
- Check whether reset pin is valid, and check whether there are multi function definition or clash in reset pin.
- Check i2c pin. Open the Camera app, and search the key word “sensor_detect” within serial port log, if "sensor_detect_xxx" can be found, which means that i2c works normally, otherwise, i2c works abnormally. If i2c address is correctly listed in the software, it needs to use oscilloscope to check whether mclk and reset is proper based on hardward concern.
- The clock for sensor. Please do configuration based on the exportable clock frequency of platform pin, and if the max frequency of pin is exceed, please change the setting.
- Check mipi clk, for clka is designed as default in the w400 board's SchematIC, if clkb is replaced, please change the below 3 registers within "subdev/sensor/src/platform/system_am_mipi.c" function.

```
  hi_csi_phy_cntl3 = f002
  mipi_phy_mux_ctrl0 = 10123
  mipi_phy_mux_ctrl0 = 10123
```

If "sensor clk", "vdd", "reset", "i2c", "mipi clk" are all checked to be ok, please use "v4l2_test" test program for verification. If it still fails with v4l2_test, please contact the module factory to check the sensor.

4.2 Debug Commands

- After burnning the board image, the log of Camera app is not appeared.

```
/* Execute the below command and the log of Camera app will be appeared */
pm enable com.android.camera2/com.android.camera.CameraLauncher
```

- The board was started normally, but the log of Camera app is not appeared.
Check whether " /dev/video50" node existed. If the node is not found, it means there are some errors issued when isp driver started. Please check the error reason.
- Check isp interruption

```
/* When isp is running, interruptions are executed always. Below command can be used to check whether isp triggers the interruption */
cat /proc/interrupts
```

- Print Camera info

```
/* Print Camera information */
dumpsys media.camera > camera.dump
```

- Read mipi's status

```
/*
  When there is no data in Camera and no initial errors found. Please read
MIPI status to check whether there is incoming data. Only 0x26 is found in
the read stream, it can be confirmed that mipi data exists.
*/
echo 0xff650050 5 > /sys/kernel/debug/aml_reg/dump
cat /sys/kernel/debug/aml_reg/dump
```

- Modify isp Register

  By modifing the SDK's source code or the isp register online, isp register can be modified. E.g.

```
  /*
      The registers from v4l2_dev/src/fw_lib/isp_config_seq.h are all isp
registers.
      If it needs to be modified permanently, it is enough to modify the
array values in the file direcrly.
      If only online debug is needed, please execute the below commands.
0x19348 is register address, 0x0000001c is register value.
  */
  open sinter
  echo w 0x19348 0x0000001c > /sys/devices/platform/fa000000.isp/reg
  close sinter
  echo w 0x19348 0x0000000c > /sys/devices/platform/fa000000.isp/reg
```

- mipi clk mode

  "mipi clk" has consistent mode and inconsistent mode. For consistent mode is adopted as default, please the mipi's input source should be configured as consistent mode.

```
  /* In default, "3d8" represents consistent mode. */
  mipi_phy1_reg_wr(MIPI_PHY_CLK_LANE_CTRL ,0x3d8);
  /* Inconsistent mode */
  mipi_phy1_reg_wr(MIPI_PHY_CLK_LANE_CTRL ,0xd8);
```

4.3 v4l2_test

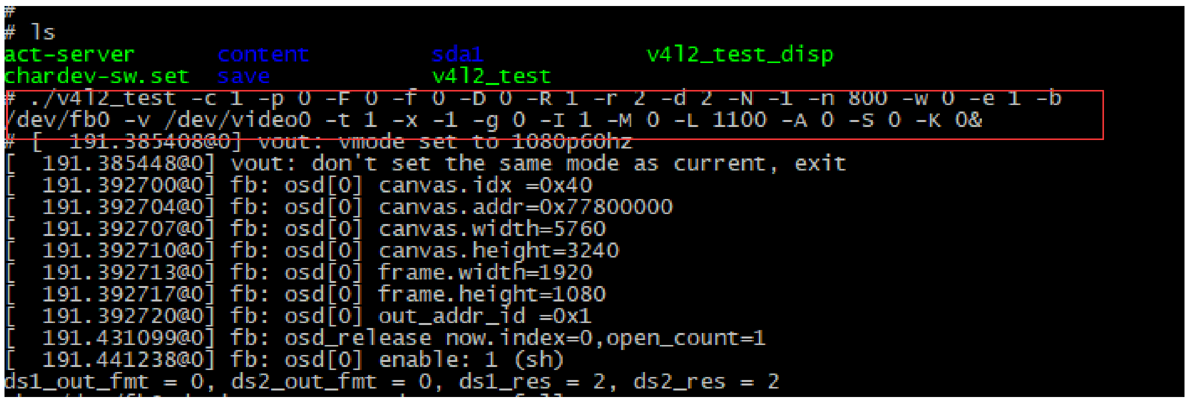
Based on v4l2 and framebuffer framework, v4l2_test is designed as an application for users to debug Camera. Users can use v4l2_test command to realize these functions, such as image display, parameter setting, etc...

4.3.1 Compiling & Utilizing v4l2_test

Enter into "hardware/aml-xxx/amlogic/arm_isp/v4l2_testapp" and execute “make”, one "v4l2_test" file will be generated in "out/target/product/***/vendor/bin" folder. It needs to re-compile for each v4l2_test.c updates.

Place the generated v4l2_test to the device's "/media" and execute v4l2_test command in "/media", debug can be started.

```
./v4l2_test -c 1 -p 0 -F 1 -f 0 -D 0 -R 1 -r 1 -d 2 -N 200 -n 200 -w 0 -e 1  -v
/dev/video50 -t 1 -x 0 -g 0 -I 1 -M 0 -L 10 -A 64 -S 0 -K 0
```

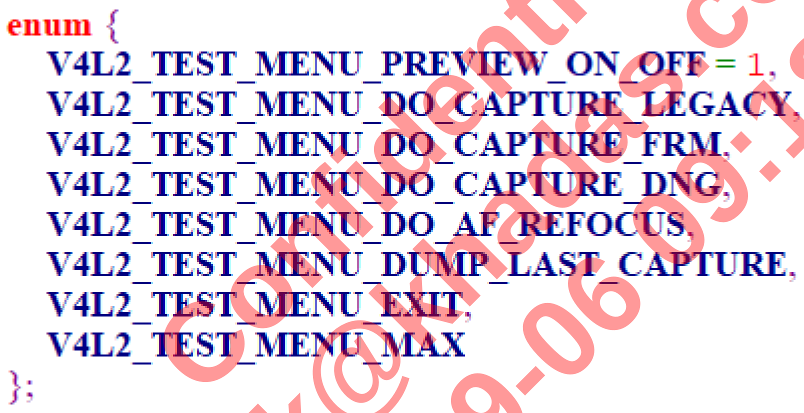


```
#
# ls
act-server       content          sda1             v4l2_test_disp
chardev-sw.set   save             v4l2_test
# ./v4l2_test -c 1 -p 0 -F 0 -f 0 -D 0 -R 1 -r 2 -d 2 -N -1 -n 800 -w 0 -e 1 -b
/dev/fb0 -v /dev/video0 -t 1 -x -1 -g 0 -I 1 -M 0 -L 1100 -A 0 -S 0 -K 0&
# [  191.385408@0] vout: vmode set to 1080p60hz
[  191.385448@0] vout: don't set the same mode as current, exit
[  191.392700@0] fb: osd[0] canvas.idx =0x40
[  191.392704@0] fb: osd[0] canvas.addr=0x77800000
[  191.392707@0] fb: osd[0] canvas.width=5760
[  191.392710@0] fb: osd[0] canvas.height=3240
[  191.392713@0] fb: osd[0] frame.width=1920
[  191.392717@0] fb: osd[0] frame.height=1080
[  191.392720@0] fb: osd[0] out_addr_id =0x1
[  191.431099@0] fb: osd_release now.index=0,open_count=1
[  191.441238@0] fb: osd[0] enable: 1 (sh)
ds1_out_fmt = 0, ds2_out_fmt = 0, ds1_res = 2, ds2_res = 2
```

4.3.2 v4l2_test Command Introduction

- -c

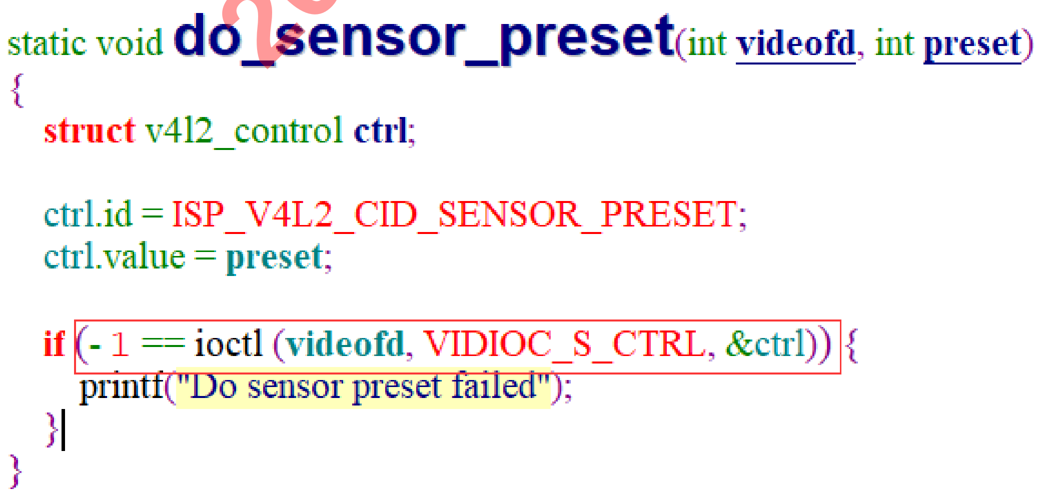
  It is relevant with capture. The parameter value is set for "command" variable, and this parameter is aligned with the enum values defined in main() function of v4l2_test.c. As shown below:



```
enum {
    V4L2_TEST_MENU_PREVIEW_ON_OFF = 1,
    V4L2_TEST_MENU_DO_CAPTURE_LEGACY,
    V4L2_TEST_MENU_DO_CAPTURE_FRM,
    V4L2_TEST_MENU_DO_CAPTURE_DNG,
    V4L2_TEST_MENU_DO_AF_REFOCUS,
    V4L2_TEST_MENU_DUMP_LAST_CAPTURE,
    V4L2_TEST_MENU_EXIT,
    V4L2_TEST_MENU_MAX
};
```

- -p

  The parameter value is set for "sensor_preset" variable, and default value is 0. Please set "sensor_preset" value with the ioctl of the interface function "do_sensor_preset(int videofd, int preset)". As shown below:



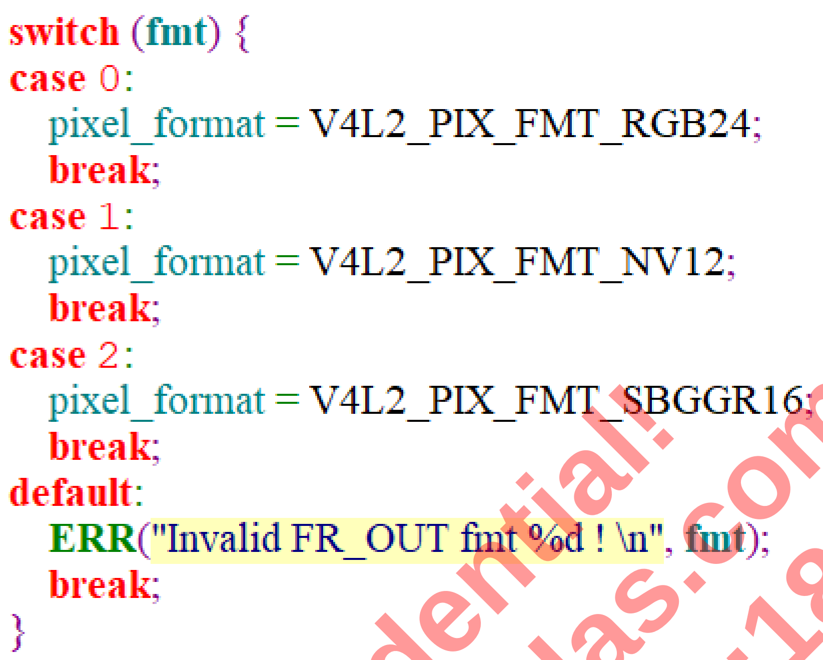
```
static void do_sensor_preset(int videofd, int preset)
{
    struct v4l2_control ctrl;

    ctrl.id = ISP_V4L2_CID_SENSOR_PRESET;
    ctrl.value = preset;

    if (- 1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
        printf("Do sensor preset failed");
    }
}
```

- -F

The parameter value is set for "fr_out_fmt" variable, and default value is 0. Three types are supported now:

0 V4L2_PIX_FMT_RGB24   /* "24-bit RGB 8-8-8" /
1 V4L2_PIX_FMT_NV12   / "Y/CbCr 4:2:0" /
3 V4L2_PIX_FMT_SBGGR16 / "RAW 16" */

If it needs to modify or add other format, please do modification at parse_fmt_res() in v4l2_test.c . As shown below:



```
switch (fmt) {
case 0:
    pixel_format = V4L2_PIX_FMT_RGB24;
    break;
case 1:
    pixel_format = V4L2_PIX_FMT_NV12;
    break;
case 2:
    pixel_format = V4L2_PIX_FMT_SBGGR16;
    break;
default:
    ERR("Invalid FR_OUT fmt %d ! \n", fmt);
    break;
}
```

- -f

  The parameter value is set for "ds1_out_fmt" variable, and default value is 0. Two types are supported now:

  0 V4L2_PIX_FMT_RGB24   /* "24-bit RGB 8-8-8" */

  1 V4L2_PIX_FMT_NV12   /* "Y/CbCr 4:2:0" */

- -D

  The parameter value is set for "ds2_out_fmt" variable, which is same as "-f" parameter.

- -R

  The parameter value is set for "fr_res" variable to set the wanted capture resolution rate.

```
switch (res) {
case 0:
    width = 3840;
    height = 2160;
    break;
case 1:
    width = 1920;
    height = 1080;
    break;
case 2:
    width = 1280;
    height = 720;
    break;
case 3:
    width = 640;
    height = 480;
    break;
default:
    ERR("Invalid resolution %d ! \n", res);
    break;
} ? end switch res ?
```

- -r

  The parameter value is set for "ds1_res" variable to capture the output resoultion from ds1 channel. Ds1 supports at most 1080p resoultion.

- -d

  The parameter value is set for "ds2_res" variable to capture the output resoultion from ds2 channel. Ds2 supports at most 1080p resoultion.

- -N

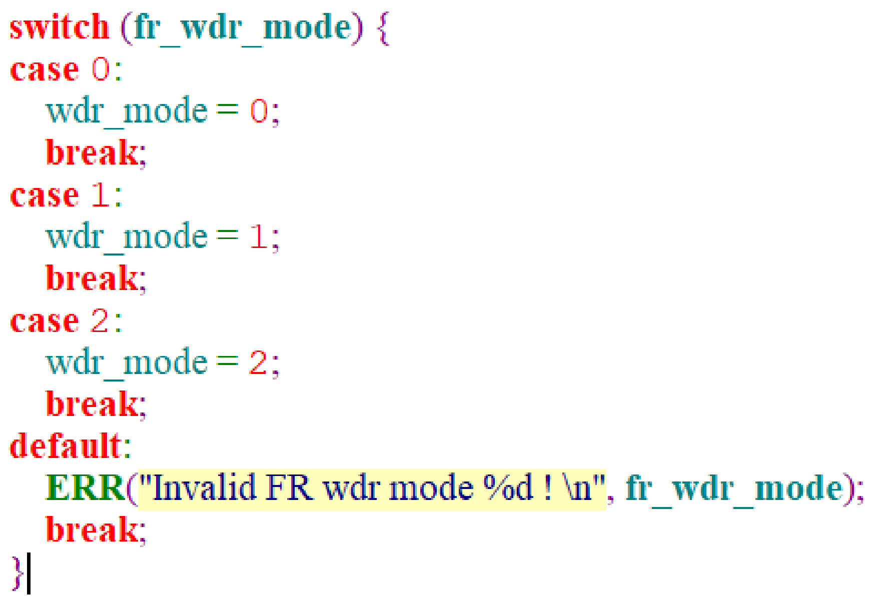
  The parameter value is set for "fr_num" variable, which sets the frame number to be captured by fr channel.

- -n

  The parameter value is set for "ds_num" variable, which sets the frame number to be captured by ds1 channel.

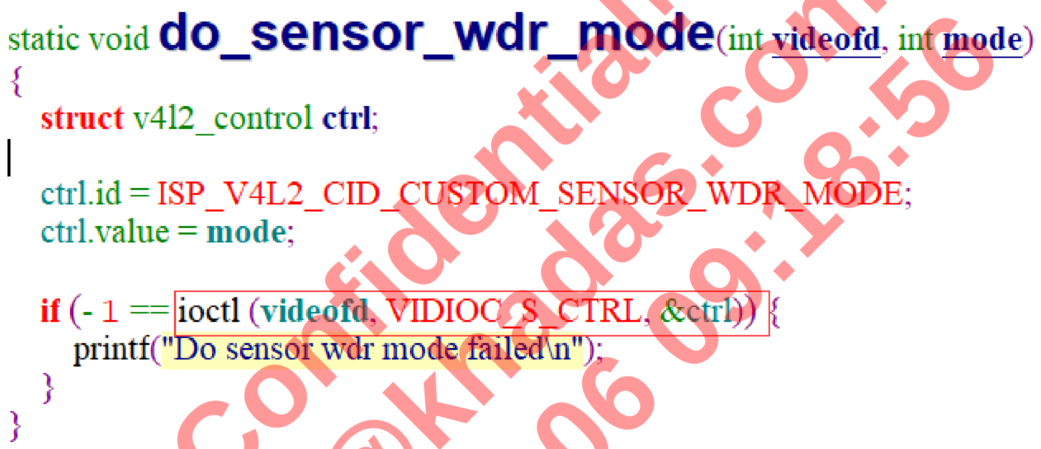
- -w

  The parameter value is set for "wdr_mode" variable, and default value is 0. Three types are supported now: 0: linear 1: native 2: fs lin. The parameter is related with fr_wdr_mode of parse_fmt_res(), as shown below:



```
switch (fr_wdr_mode) {
case 0:
    wdr_mode = 0;
    break;
case 1:
    wdr_mode = 1;
    break;
case 2:
    wdr_mode = 2;
    break;
default:
    ERR("Invalid FR wdr mode %d ! \n", fr_wdr_mode);
    break;
}
```

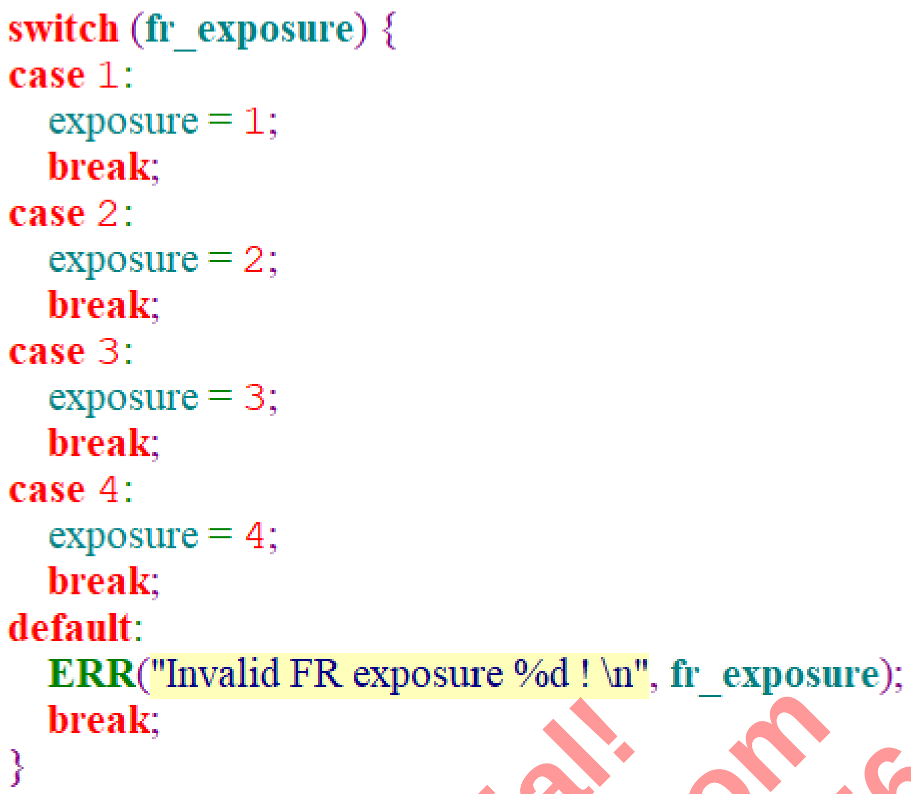
Please set "wdr_mode" value with the ioctl of the interface function "do_sensor_wdr_mode(int videofd, int mode)", as shown below:



```
static void do_sensor_wdr_mode(int videofd, int mode)
{
    struct v4l2_control ctrl;

    ctrl.id = ISP_V4L2_CID_CUSTOM_SENSOR_WDR_MODE;
    ctrl.value = mode;

    if (-1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
        printf("Do sensor wdr mode failed\n");
    }
}
```

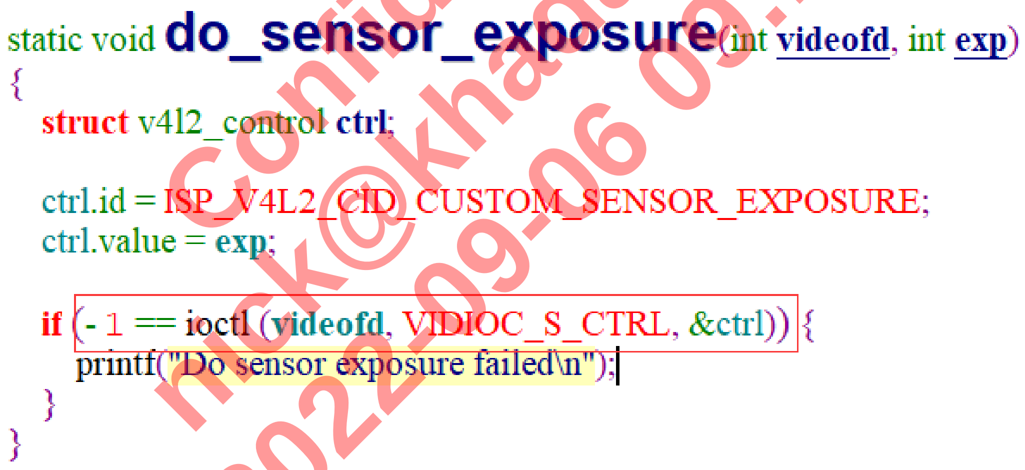
- -e

  The parameter value is set for "exposure" variable and just as exposure value. The default value is 1. Four types are supported now. The parameter is related with fr_exposure of parse_fmt_res(), as shown below:



```
switch (fr_exposure) {
case 1:
    exposure = 1;
    break;
case 2:
    exposure = 2;
    break;
case 3:
    exposure = 3;
    break;
case 4:
    exposure = 4;
    break;
default:
    ERR("Invalid FR exposure %d ! \n", fr_exposure);
    break;
}
```

Please set "exposure" value with the ioctl of the interface function "do_sensor_exposure(int videofd, int exp)", as shown below:



```
static void do_sensor_exposure(int videofd, int exp)
{
    struct v4l2_control ctrl;

    ctrl.id = ISP_V4L2_CID_CUSTOM_SENSOR_EXPOSURE;
    ctrl.value = exp;

    if (-1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
      printf("Do sensor exposure failed\n");
    }
}
```

- -b

  The parameter value is set for "fbdevname" variable and just as "fb dev name". The default value is "/dev/fb0".

- -v

  The parameter value is set for "v4ldevname" variable and just as "video dev name". The default value is "/dev/video0".

- -t

  The parameter value is set for "open_port_cnt" variable and just as "run the port count". The default value is "1".

- -x

  The parameter value is set for "fps_test_port" variable and just as "fps printf port". Totally there are 4 types: -1, no printf; 0, fr; 1, meta; 2, ds1; 3, ds2. The default value is -1, no printf.

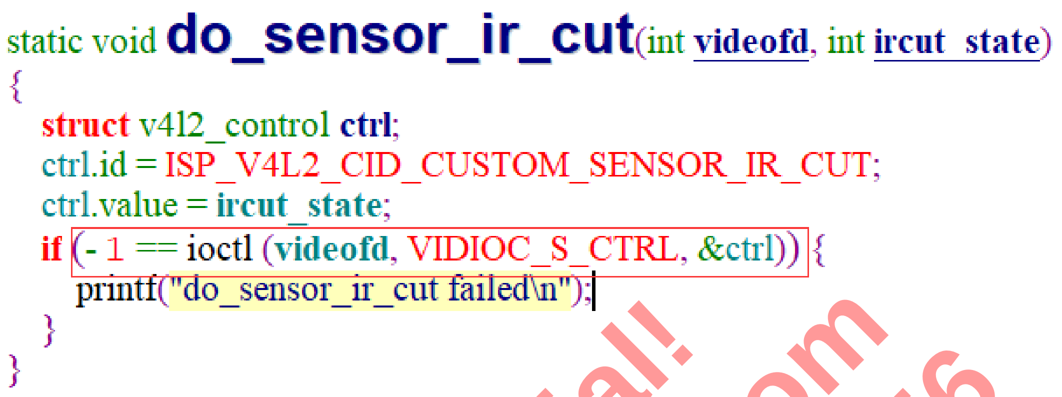
- -g

  The parameter value is set for "ds_gdc_ctrl" variable and just as "enable or disable gdc module". Totally there are 2 types: 0, disable; 1, enable. The default value is 0.

- -I

  The parameter value is set for "ir_cut_state" variable and just as "eset sensor ir cut state". Totally there are 2 types: 0, close; 1, open. The default value is 1.

  Please set "ir_cut_state" value with the ioctl of the interface function "do_sensor_ir_cut(int videofd, int ircut_state)", as shown below:

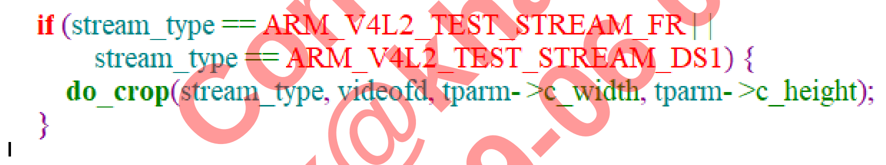


```
static void do_sensor_ir_cut(int videofd, int ircut_state)
{
    struct v4l2_control ctrl;
    ctrl.id = ISP_V4L2_CID_CUSTOM_SENSOR_IR_CUT;
    ctrl.value = ircut_state;
    if (- 1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
        printf("do_sensor_ir_cut failed\n");
    }
}
```

- -W

  The parameter value is set for "fr_c_width" variable and just as "fr crop width". The default value is 0.

  Please set "fr_c_width" value with the ioctl of the interface function "do_crop(int type, int videofd, int width, int height)", as shown below:



```
if (stream_type == ARM_V4L2_TEST_STREAM_FR ||
    stream_type == ARM_V4L2_TEST_STREAM_DS1) {
  do_crop(stream_type, videofd, tparm->c_width, tparm->c_height);
}
```

- -H

  The parameter value is set for "fr_c_height" variable and just as "fr crop height". The default value is 0.

- -Y

  The parameter value is set for "ds_c_width" variable and just as "ds1 crop width". The default value is 0.

- -Z

  The parameter value is set for "ds_c_height" variable and just as "ds1 crop height". The default value is 0.

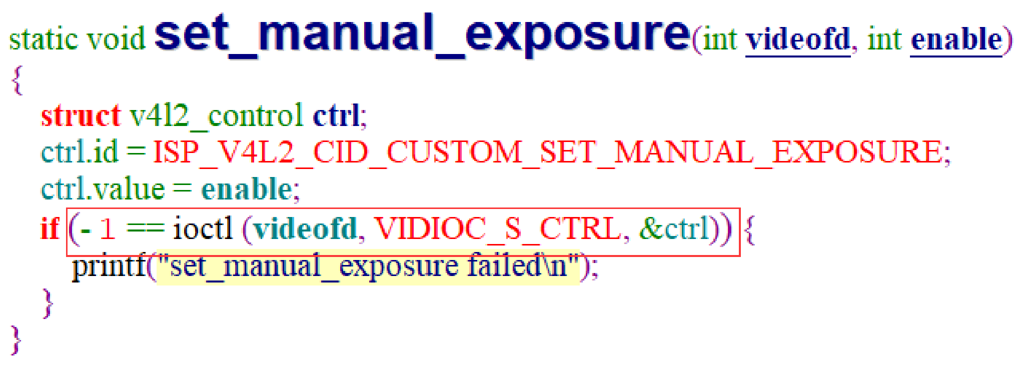
- -a

  The parameter value is set for "fr_a_ctrl" variable and just as "fr zone weight ctrl". The default value is 0.

- -M

  The parameter value is set for "manual_exposure_enable" variable and just as "enable or disable manual exposure". Totally there are 2 types: 0, disable; 1, enable. The default value is 0.

  Please set "manual_exposure_enable" value with the ioctl of the interface function "set_manual_exposure(int videofd, int enable)", as shown below:

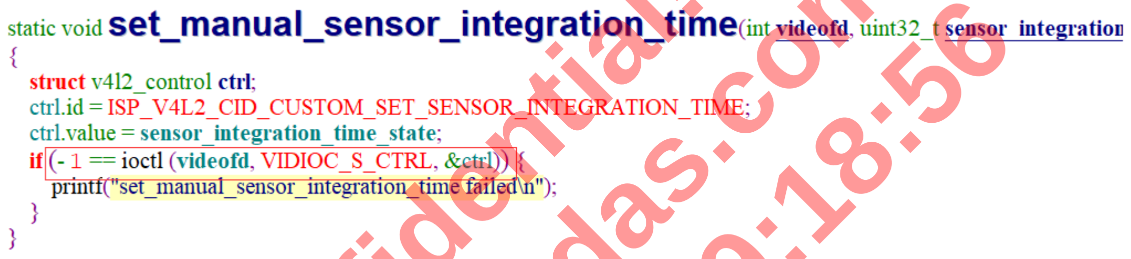


```
static void set_manual_exposure(int videofd, int enable)
{
    struct v4l2_control ctrl;
    ctrl.id = ISP_V4L2_CID_CUSTOM_SET_MANUAL_EXPOSURE;
    ctrl.value = enable;
    if (- 1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
        printf("set_manual_exposure failed\n");
    }
}
```

- -L

  The parameter value is set for "manual_sensor_integration_time" variable and just as "set integration time". The default value is 1.

  Please set "manual_sensor_integration_time" value with the ioctl of the interface function "set_manual_sensor_integration_time(int videofd, uint32_t sensor_integration_time_state)", as shown below:

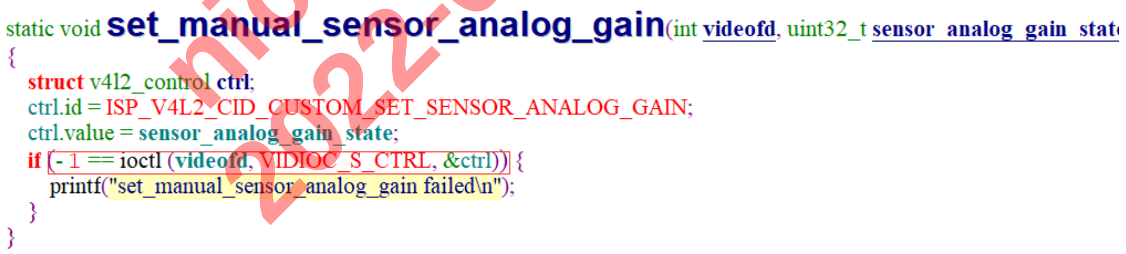


```
static void set_manual_sensor_integration_time(int videofd, uint32_t sensor_integration
{
    struct v4l2_control ctrl;
    ctrl.id = ISP_V4L2_CID_CUSTOM_SET_SENSOR_INTEGRATION_TIME;
    ctrl.value = sensor_integration_time_state;
    if (- 1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
        printf("set_manual_sensor_integration_time failed\n");
    }
}
```

- -A

  The parameter value is set for "manual_sensor_analog_gain" variable and just as "set analog gain". The default value is 0.

  Please set "manual_sensor_analog_gain" value with the ioctl of the interface function "set_manual_sensor_analog_gain(int videofd, uint32_t sensor_analog_gain_state)", as shown below:

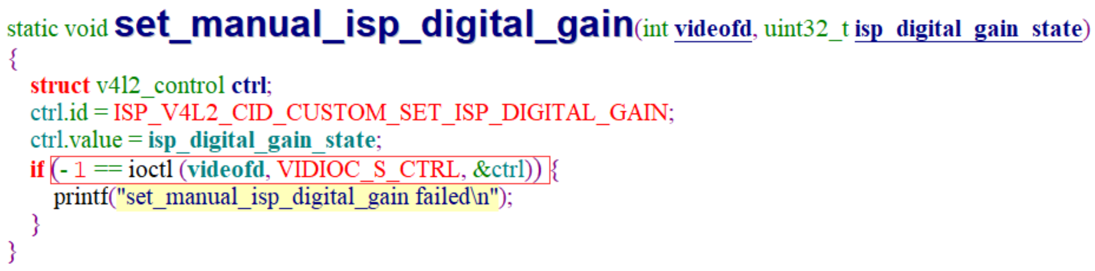


```
static void set_manual_sensor_analog_gain(int videofd, uint32_t sensor_analog_gain_stat
{
    struct v4l2_control ctrl;
    ctrl.id = ISP_V4L2_CID_CUSTOM_SET_SENSOR_ANALOG_GAIN;
    ctrl.value = sensor_analog_gain_state;
    if (- 1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
        printf("set_manual_sensor_analog_gain failed\n");
    }
}
```

- -S

  The parameter value is set for "manual_isp_digital_gain" variable and just as "manual_isp_digital_gain". The default value is 0.

  Please set "manual_isp_digital_gain" value with the ioctl of the interface function "set_manual_isp_digital_gain(int videofd, uint32_t isp_digital_gain_state)", as shown below:

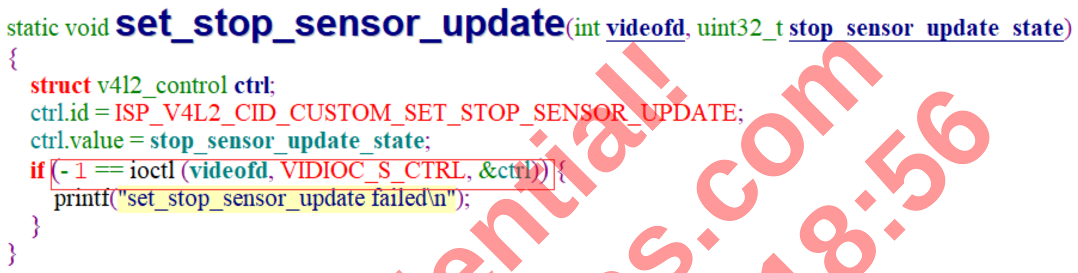


```
static void set_manual_isp_digital_gain(int videofd, uint32_t isp_digital_gain_state)
{
   struct v4l2_control ctrl;
   ctrl.id = ISP_V4L2_CID_CUSTOM_SET_ISP_DIGITAL_GAIN;
   ctrl.value = isp_digital_gain_state;
   if (- 1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
      printf("set_manual_isp_digital_gain failed\n");
   }
}
```

- -K

  The parameter value is set for "stop_sensor_update" variable and just as "stop sensor update". Totally there are 2 types: 0, enable sensor update, 1, stop sensor update. The default value is 0.

  Please set "stop_sensor_update" value with the ioctl of the interface function "set_stop_sensor_update(int videofd, uint32_t stop_sensor_update_state)", as shown below:



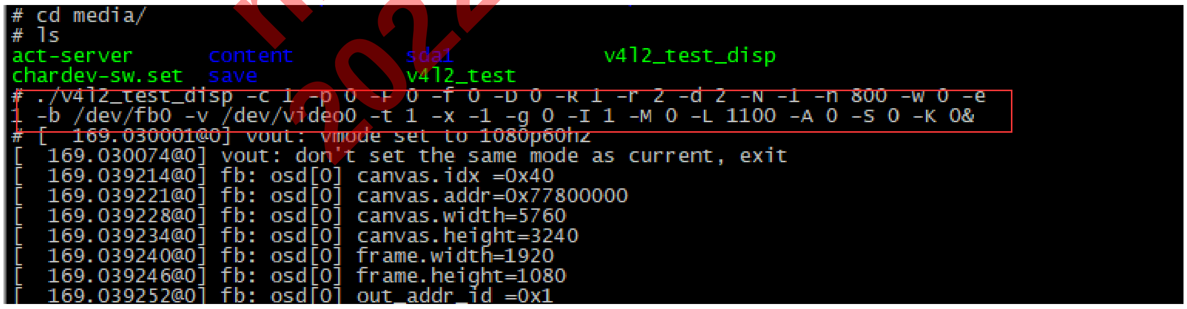
```
static void set_stop_sensor_update(int videofd, uint32_t stop_sensor_update_state)
{
   struct v4l2_control ctrl;
   ctrl.id = ISP_V4L2_CID_CUSTOM_SET_STOP_SENSOR_UPDATE;
   ctrl.value = stop_sensor_update_state;
   if (- 1 == ioctl (videofd, VIDIOC_S_CTRL, &ctrl)) {
      printf("set_stop_sensor_update failed\n");
   }
}
```

4.3.3 v4l2_test Common Examples

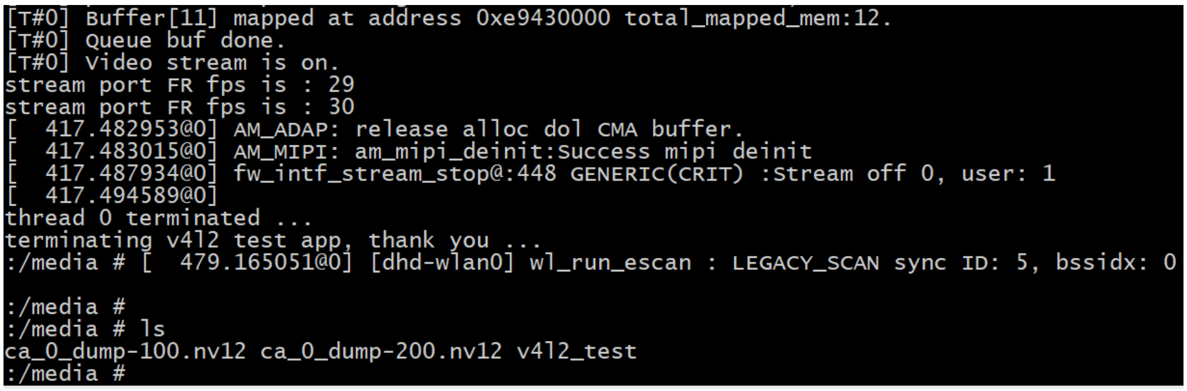
Utilize fr channel to display 1080p

```
./v4l2_test -c 1 -p 0 -F 0 -f 0 -D 0 -R 1 -r 2 -d 2 -N -1 -n 800 -w 0 -e 1 -b
/dev/fb0 -v /dev/video0 -t 1 -x -1 -g 0 -I 1 -M 0 -L 1100 -A 1 -S 0 -K 0&
```

Please refer the example listed below:



```
# cd media/
# ls
act-server      content         sda1            v4l2_test_disp
chardev-sw.set  save            v4l2_test
# ./v4l2_test_disp -c 1 -p 0 -F 0 -f 0 -D 0 -R 1 -r 2 -d 2 -N -1 -n 800 -w 0 -e
1 -b /dev/fb0 -v /dev/video0 -t 1 -x -1 -g 0 -I 1 -M 0 -L 1100 -A 0 -S 0 -K 0&
# [  169.030001@0] vout: vmode set to 1080p60hz
[  169.030074@0] vout: don't set the same mode as current, exit
[  169.039214@0] fb: osd[0] canvas.idx =0x40
[  169.039221@0] fb: osd[0] canvas.addr=0x77800000
[  169.039228@0] fb: osd[0] canvas.width=5760
[  169.039234@0] fb: osd[0] canvas.height=3240
[  169.039240@0] fb: osd[0] frame.width=1920
[  169.039246@0] fb: osd[0] frame.height=1080
[  169.039252@0] fb: osd[0] out_addr_id =0x1
```



```
[T#0] Buffer[11] mapped at address 0xe9430000 total_mapped_mem:12.
[T#0] Queue buf done.
[T#0] Video stream is on.
stream port FR fps is : 29
stream port FR fps is : 30
[  417.482953@0] AM_ADAP: release alloc dol CMA buffer.
[  417.483015@0] AM_MIPI: am_mipi_deinit:Success mipi deinit
[  417.487934@0] fw_intf_stream_stop@:448 GENERIC(CRIT) :Stream off 0, user: 1
[  417.494589@0]
thread 0 terminated ...
terminating v4l2 test app, thank you ...
:/media # [  479.165051@0] [dhd-wlan0] wl_run_escan : LEGACY_SCAN sync ID: 5, bssidx: 0

:/media #
:/media # ls
ca_0_dump-100.nv12 ca_0_dump-200.nv12 v4l2_test
:/media #
```

When "fps" is printed, it means that the data capture is succeed. It is possible to check the captured files under current folder.