TCCxxx Android

Camera Module Porting Guide

Android_ALL_V1. 50_Camera Module Porting Guide

Feb. 13, 2014.

Telechips

DISCLAIMER

All information and data contained in this material are without any commitment, are not to be considered as an offer for conclusion of a contract, nor shall they be construed as to create any liability. Any new issue of this material invalidates previous issues. Product availability and delivery are exclusively subject to our respective order confirmation form; the same applies to orders based on development samples delivered. By this publication, Telechips, Inc. does not assume responsibility for patent infringements or other rights of third parties that may result from its use.

Further, Telechips, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.
No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of Telechips, Inc.
This product is designed for general purpose, and accordingly customer be responsible for all or any of intellectual property licenses required for actual application. Telechips, Inc. does not provide any indemnification for any intellectual properties owned by third party.
Telechips, Inc. can not ensure that this application is the proper and sufficient one for any other purposes but the one explicitly expressed herein. Telechips, Inc. is not responsible for any special, indirect, incidental or consequential damage or loss whatsoever resulting from the use of this application for other purposes.

COPYRIGHT STATEMENT

Important Notice

For customers who use licensed Codec ICs and/or licensed codec firmware of mp3:
"Supply of this product does not convey a license nor imply any right to distribute content created with this product in revenue-generating broadcast systems (terrestrial. Satellite, cable and/or other distribution channels), streaming applications(via internet, intranets and/or other networks), other content distribution systems(pay-audio or audio-on-demand applications and the like) or on physical media(compact discs, digital versatile discs, semiconductor chips, hard drives, memory cards and the like). An independent license for such use is required. For details, please visit http://mp3licensing.com".

For customers who use other firmware of mp3:
"Supply of this product does not convey a license under the relevant intellectual property of Thomson and/or Fraunhofer Gesellschaft nor imply any right to use this product in any finished end user or ready-to-use final product. An independent license for such use is required. For details, please visit http://mp3licensing.com".

For customers who use Digital Wave DRA solution:
"Supply of this implementation of DRA technology does not convey a license nor imply any right to this implementation in any finished end-user or ready-to-use terminal product. An independent license for such use is required."

For customers who use DTS technology:
"This product made under license to certain U.S. patents and/or foreign counterparts."
"© 1996 – 2011 DTS, Inc. All rights reserved."

For customers who use Dolby technology:
"Supply of this Implementation of Dolby technology does not convey a license nor imply a right under any patent, or any other industrial or intellectual property right of Dolby Laboratories, to use this Implementation in any finished end-user or ready-to-use final product. It is hereby notified that a license for such use is required from Dolby Laboratories."

For customers who use MS technology:
"This product is subject to certain intellectual property rights of Microsoft and cannot be used or disctributed further without the appropriate license(s) from Microsoft."

1. Revision History

Date	Version	Description
2010-02-01	1.00	Initial release
2010-04-05	1.10	Addition sony-5MP sensor module
2011-03-08	1.20	Addition QnA
2011-03-15	1.21	Modification QnA (5.2, 6.4, 6.11, 6.12)
2011-04-01	1.30	Edition for Gingerbread version
2011-04-08	1.31	Addition The Changed circuit for ISP Interface(4.7)
2012-05-18	1.32	Add How to setting DE signal and HS signal for TCC892x
2013-02-07	1.40	Edition for JB-mr1 version.
2014-02-10	1.50	Edition for Kitkat version.

2. TABLE OF CONTENTS

2.1. Contents

2.2. Tables

3. Overview

This documentation explains default camera module which is supported by Telechips Android platform and describe how to add new camera module in Android platform which was ported by Telechips.

3.1. Supports Camera modules

1) Dual Camera

Module name	Pixels	Source name
MT9P111	5MP	mt9p111_5mp.c (h)
MT9M113	1.3MP	mt9m113.c (h)
S5K5CAGA	3MP	s5k5caga_3mp.c(h)
SR130PC10	1.3MP	sr130pc10_1.3mp.c(h)

Table 1. Dual Camera Module list

2) Single Camera

Module name	Pixels	Source name
MT9D112	2MP	mt9d112_2mp.c (h)
S5K4BAFB	2MP	s5k4bafb_2mp.c (h)
MT9T111	3MP	mt9t111_3mp.c (h)
MT9P111	5MP	mt9p111_5mp.c (h)
MT9M113	1.3MP	Mt9m113.c (h)

Table 2. Single Camera Module list

Function name	Discription
Write_regs	I2c function to send Camera module command data.
Sensor_open	Power-up sequence. - Power up, reset, Clock enable. Value setting for sensor initial code.
sensor_close	Power-down sequence. - Power down, Clock disable.
sensor_preview	Value setting for preview mode.
sensor_capture	Value setting for capture mode.
sensor_capturecfg	Pre-process for capture.
sensor_zoom	Value setting for zoom.
sensor_autofocus	Value setting for AutoFocus.
sensor_effect	Value setting for Color Effect.
sensor_flip	Value setting for Mirror/Flip.
sensor_iso	Value setting for ISO.
sensor_me	Value setting for Metering Exposure.
sensor_wb	Value setting for White Balance.
sensor_scene	Value setting for Scene mode.
sensor_exposure	Value setting for Exposure.
sensor_check_esd	To check ESD.

Table 3. Camera Function list

Source path : Android_home\kernel\drivers\media\video\tcccam*.*

4. Add new module in kernel

This section explains how to add new camera sensors
Please refer to below items to add new camera module in android platform which was ported by Telechips.

4.1. And new driver module

- 4.1.1 Add I2C Slave Address

To set of I2C slave address, source paths are as follows

In case of TCC892x : Android_home\kernel/arch/arm/mach-tcc892x/board-tcc8920-camera.c
In case of TCC893x : Android_home\kernel/arch/arm/mach-tcc892x/board-tcc8930-camera.c

- 4.1.1.1 For Single camera

In Table 4, the yellow shaded texts are the feature of camera module that is selected by customer.
For example, this is CONFIG_VIDEO_CAMERA_SENSOR_MT9P111.
The model name of camera sensor module is MT9P111 and it's made define in *Kconfig file.
You have to put the I2C slave address of camera module that is selected by customer as SENSOR_I2C_SLVAE_ID.

* Kconfig : refer to 4.4.1

Android_home\kernel/arch/arm/mach-tcc893x/board-tcc8930-camera.c

```
#if defined(CONFIG_VIDEO_TCCXX_CAMERA)
#include <media/cam_i2c.h>
#if defined(CONFIG_VIDEO_CAMERA_SENSOR_AIT848_ISP)
#define SENSOR_I2C_SLAVE_ID             (0x06>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9P111)
#define SENSOR_I2C_SLAVE_ID             (0x7A>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_MV9317)
#define SENSOR_I2C_SLAVE_ID             (0x50>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9D112)
#define SENSOR_I2C_SLAVE_ID             (0x7A>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_OV3640)
#define SENSOR_I2C_SLAVE_ID             (0x78>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_S5K4BAFB)
#define SENSOR_I2C_SLAVE_ID             (0x52>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_ISX006)
#define SENSOR_I2C_SLAVE_ID             (0x34>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_OV7690)
#define SENSOR_I2C_SLAVE_ID             (0x42>>1)
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9M113)
#define SENSOR_I2C_SLAVE_ID             (0x78>>1)
#elif defined(CONFIG_VIDEO_ATV_SENSOR_TVP5150)
#define SENSOR_I2C_SLAVE_ID             (0xB8>>1)
#elif defined(CONFIG_VIDEO_ATV_SENSOR_RDA5888)
#define SENSOR_I2C_SLAVE_ID             (0xC4>>1)
#elif defined(CONFIG_VIDEO_CAMERA_NEXTCHIP_TEST)
#define SENSOR_I2C_SLAVE_ID             (0x50>>1)
#endif
#endif // defined(CONFIG_VIDEO_TCCXX_CAMERA)
```

Table 4. I2C Slave Address For Single camera

- 4.1.1.2 For Dual camera

To use dual camera, the green shaded texts of i2c_devices1[] array in Table.5 have to changed with the I2C slave address of camera module that is selected by customer.

Android_home\kernel/arch/arm/mach-tcc893x/board-tcc8930.c

```
static struct i2c_board_info __initdata i2c_camera_devices [] = {
        #if defined(CONFIG_VIDEO_DUAL_CAMERA_SUPPORT)
        {
                #if defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9P111)
                        I2C_BOARD_INFO("tcc-cam-sensor-0", (0x7A>>1)),    // For Back Camera
                #endif
                .platform_data = &cam_i2c_data1,
        },
        {
                #if defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9M113)
                        I2C_BOARD_INFO("tcc-cam-sensor-1", (0x78>>1)),    // For Front Camera
                #endif
                .platform_data = &cam_i2c_data1,
        },
        #else // CONFIG_VIDEO_DUAL_CAMERA_SUPPORT
```

Table 5. I2C Slave Address For Dual camera

- 4.1.2 Make I2C write/read function.

- 4.1.2.1 Make Sensor Initialization Code Table

Initialization code has a different command type according to camera sensor module.
Refer to Table 6.
The command of I2C is written on register of camera sensor and has the variable length.
For example, length is 1byte, 2byte or more.

Android_home\kernel/drivers/media/video/tcccam/mt9p111_5mp.c(h)

```
enum sensor_reg_width {
        WORD_LEN,
        BYTE_LEN,
        MS_DELAY
};

struct sensor_reg {
        unsigned short reg;
        unsigned short val;
        enum sensor_reg_width width;
};

static struct sensor_reg sensor_initialize_mt9p111[] = {
// Begin_Initialize
{ 0x0010, 0x0340 , WORD_LEN},
{ 0x0012, 0x0090 , WORD_LEN},    // in case 2 byte command
{ 0x0014, 0x2025 , WORD_LEN},
{ 0x001E, 0x0665 , WORD_LEN},
{ 0x0022, 0x0030 , WORD_LEN},
{ 0x002A, 0x7F7F , WORD_LEN},
{ 0x002C, 0x0000 , WORD_LEN},
{ 0x002E, 0x0000 , WORD_LEN},
{ 0x0018, 0x4008 , WORD_LEN},
```

```
{MT9P111_REG_TERM, 0x000A, WORD_LEN},    // delay code, 10 ms of units , Hex type.

…
…
{ 0xB040, 0x01     , BYTE_LEN},    // in case 1 byte command
{ 0x8404, 0x06     , BYTE_LEN}
…
…

{MT9P111_REG_TERM, MT9P111_VAL_TERM, WORD_LEN}    // Termination Code
};
```

Table 6. Camera Sensor module Initialization Code

- 4.1.2.2 Make I2C Write/Read Function

Depending on 4.1.2.1, you must make the I2C write function
The write_regs_mt9p111() in Table 7 is the function that write Initialization code table of MT9P111 sensor module as I2C command.

Android_home\kernel/drivers/media/video/tcccam/mt9p111_5mp.c

```
static int write_regs_mt9p111(const struct sensor_reg reglist[])
{
        int err;
        int err_cnt = 0;
        int sleep_cnt = 100;
        unsigned char data[132];
        unsigned char bytes;
        const struct sensor_reg *next = reglist;

        while (!((next->reg == MT9P111_REG_TERM) && (next->val == MT9P111_VAL_TERM)))
        // Termination Code
        {

                if(next->reg == MT9P111_REG_TERM && next->val != MT9P111_VAL_TERM)
                {
                        //mdelay(next->val);
                        msleep(next->val);    // this is delay function
                        sleep_cnt = 100;
                        printk("Sensor init Delay[%d]!!!! \n", next->val);
                        next++;
                }
                else
                {
                        sleep_cnt--;
                        if(sleep_cnt == 0)
                        {
                                msleep(10);
                                sleep_cnt = 100;
                        }

                        if(next->width == WORD_LEN){
                                bytes = 0;
                                data[bytes]= next->reg>>8;          bytes++;
                                data[bytes]= (u8)next->reg&0xff;    bytes++;
```

```
                data[bytes]= next->val>>8;          bytes++;
                data[bytes]= (u8)next->val&0xff;    bytes++;

                err = DDI_I2C_Write(data, 2, bytes-2); //4 byte Write function
                        //  total send(write) byte # = 2+bytes-2
        }
        else{
                bytes = 0;
                data[bytes]= next->reg>>8;          bytes++;
                data[bytes]= (u8)next->reg&0xff;    bytes++;

                //data[bytes]= next->val;           bytes++;
                data[bytes]= (u8)next->val&0xff;    bytes++;

                err = DDI_I2C_Write(data, 2, bytes-2); //3 byte Write function

        }
```

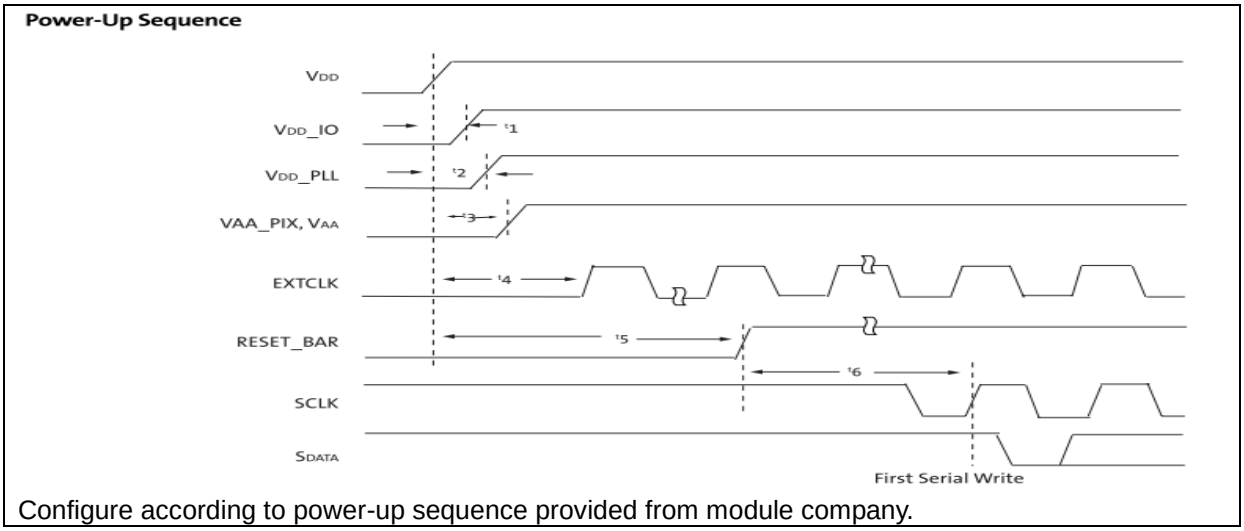
Table 7. I2C Write/Read Function

4.2. Camera GPIO Control

- 4.2.1 Adjust Power Up/Down Sequence

Before you send I2C command to the sensor module of camera, have to do the sequence of Power-Up in order to initialization of operation in camera module.

Refer to Table 8.
The sequence of Power-Up provided from a company of module.



Configure according to power-up sequence provided from module company.

Table 8. power-up sequence of Camera sensor module

The sensor_open_mt9p111() in Table 9 is made depend on Table 8, the sequence of Power-Up.
The sensor_close_mt9p111() in Table 9 is made depend on the datasheet of the camera module.
The power sequence of all must refer to the datasheet of the camera module, certainly.

Android_home\kernel/drivers/media/video/tcccam/mt9p111_5mp.c

```
static int sensor_open_mt9p111(void)
{
	sensor_power_disable();
	sensor_delay(10);

	sensor_power_enable();
	sensor_delay(10);

	sensor_powerdown_disable();
	sensor_delay(10);

	sensor_reset_low();
	sensor_delay(10);

	CIF_Open();    // EXTCLK, M clk
	sensor_delay(40);

	sensor_reset_high();
	sensor_delay(15);

	return write_regs_mt9p111(sensor_reg_common_mt9p111[0]);
```

```
}

static int sensor_close_mt9p111(void)
{
        CIF_ONOFF(OFF);

        sensor_reset_low();
        sensor_power_disable();
        sensor_powerdown_enable();

        CIF_Close();
        msleep(5);

        return 0;
}
```

Table 9. Code of sensor_open_mt9p111(), sensor_close_mt9p111()

- 4.2.2 Make GPIO Control

We recommend to use sensor module to possible Power-Down(stand-by).
please refer to 5.4-2.

1) In case of Single camera

Please refer to Table 10. “Configure GPIO function”
You have to fill in else{} with GPIO setting according to the hardware schematics.

You have to modify following functions according to real hardware schematic.
- sensor_powerdown_enable()
- sensor_powerdown_disable()
- sensor_reset_high()
- sensor_reset_low()

Android_home\kernel/driver/media/video/tcccam/sensor_if.c

```
//Configure GPIO function according to H/W configuration for camera module.

void sensor_powerdown_disable(void)
{
…
   #elif defined(CONFIG_ARCH_TCC892X) || defined(CONFIG_ARCH_TCC893X)
…
      #if defined(CONFIG_ARCH_TCC893X)
…
         else if(system_rev == 0x2000 || system_rev == 0x3000) {
            #if defined(CONFIG_VIDEO_DUAL_CAMERA_SUPPORT)
               if(CameraID) {
                  gpio_set_value(TCC_GPD(2), 0);
               } else {
                  gpio_set_value(TCC_GPG(1), 0);
               }
            #else
                  // todo : customer coding part.
                  // if you want to control the power, this part should be coding.
            #endif
…
```

```
…
}

void sensor_reset_high(void)
{
…
    #elif defined(CONFIG_ARCH_TCC892X) || defined(CONFIG_ARCH_TCC893X)
…
     #if defined(CONFIG_ARCH_TCC893X)
…
        else if(system_rev == 0x2000 || system_rev == 0x3000) {
            #if defined(CONFIG_VIDEO_DUAL_CAMERA_SUPPORT)
                if(CameraID) {
                    gpio_set_value(TCC_GPF(15), 1);
                } else {
                    gpio_set_value(TCC_GPD(1), 1);
                }
            #else
                    // todo : customer coding part.
                    // if you want to control the reset, this part should be coding.
            #endif
…
}
```

Table 10. Code of GPIO control in case of Single Camera

2) In case of Dual camera

You have to refer to CONFIG_VIDEO_DUAL_CAMERA_SUPPORT, the sensor_powerdown_enable() in Table 11.
The green shaded texts, the code in if(CameraID) { }, is a setting of GPIO about a front camera.
The yellow shaded texts, the code in else { }, is a setting of GPIO about a back(rear) camera.

Android_home\kernel/driver/media/video/tcccam/sensor_if.c

```
void sensor_powerdown_enable(void)
{
…
    #elif defined(CONFIG_ARCH_TCC892X) || defined(CONFIG_ARCH_TCC893X)
…
     #if defined(CONFIG_ARCH_TCC893X)
…
        else if(system_rev == 0x2000 || system_rev == 0x3000) {
            #if defined(CONFIG_VIDEO_DUAL_CAMERA_SUPPORT)
                if(CameraID) {
                    gpio_set_value(TCC_GPD(2), 1);
                } else {
                    gpio_set_value(TCC_GPG(1), 1);
                }
            #else
                    …
                    …
            #endif
…
…
}
```

Table 11. Code of GPIO control in case of Dual Camera

About current consumption, Camera modules are bigger than the other devices. So the processing of current consumption is important issue.
Especially, for dual camera, before the one sensor open, the other sensor has to check whether it does go to power down mode. If it doesn't go to power down mode, it has to add the processing of power down.
About this matter, for dual camera, sensor_get_powerdown() function is added.

```
int sensor_get_powerdown(void)
{
…
        #elif defined(CONFIG_ARCH_TCC892X) || defined(CONFIG_ARCH_TCC893X)
…
            #if defined(CONFIG_ARCH_TCC893X)
                if(system_rev == 0x1000) {        // TCC8930
                   #if efined(CONFIG_VIDEO_DUAL_CAMERA_SUPPORT)
                      if(CameraID) {                  // front camera
                            return gpio_get_value(TCC_GPF(15));
                      } else {                        // back camera
                            return gpio_get_value(TCC_GPF(29));
                      }
…
}
```

Table 12. Code of GPIO control in case of Dual Camera

Notices : MT9P111 is consumed 4mA current when the power down mode(stand-by mode).

4.3. Fill Initialization information of new camera module

- 4.3.1 Add Kernel init information In case of Single Camera

1) Fill out Sensor header file information.

You have to use PLL1 and PLL2 to set the frequency of clock in camera device driver and have to adjust the frequency of PLL1 or PLL2 according to clock of camera module.
The clock of camera module is master clock (mclk) and CIF scaler clock.
You can get approximate value of frequency by dividing PLL1 or PLL2 by the clock of camera module.

EX)
432(PLL2)/24(m_clk) = 18 -> PLL2 is selected PLL of m_clk(master clock)
432(PLL2)/144(scaler_clk ≒ pixel clock(76) x 2) = 3 -> PLL2 is selected PLL of CIF scaler clock

To set of PLL clock, source paths are as follows

In case of TCC892x, bootable/bootloader/lk/target/tcc892x_evm/clock.c
In case of TCC893x, bootable/bootloader/lk/target/tcc893x_evm/clock.c

The clock_init_early() in Table13 initialize the frequency of PLL clocks.

Android_home\bootable/bootloader/lk/target/ tcc893x_evm/clock.c

```
void clock_init_early (void)
{
…
…
#if defined(CONFIG_AUDIO_PLL_USE)
        tca_ckc_setpll( 3750000, 3, PLLSRC_XIN);
#else
        tca_ckc_setpll( 7680000, 3, PLLSRC_XIN);
#endif
        tca_ckc_setpll( 5940000, 4, PLLSRC_XIN);…
…

}
```

Table 13. Example of Camera clocks setting

There is a part of header file about camera driver in table14 and it is important definitions in header file.
You have to modify the yellow shaded texts according to each camera module.
Please refer notes of Table14.

Android_home\kernel/driver/media/video/tcccam/mt9p111_5mp.h

```
#if defined(CONFIG_ARCH_TCC892X) || defined(CONFIG_ARCH_TCC893X)
#if defined(CONFIG_USE_ISP)
#define CAM_POLARITY_VSYNC  0    // 1: low active,        0: high active
#define CAM_POLARITY_HSYNC  0    // 1: low active,        0: high active
#define CAM_POLARITY_PCLK   0    // 1: positive edge      0: negative edge
#endif //CONFIG_USE_ISP
/* notes :
Define scaler clock for MCLK and Zoom and select source PLL to make each clock
Telechips recommend to use TCC_Scaler clock which is 2 times PCLK.
*/
#define CKC_CAMERA_MCLK                                  240000
#define CKC_CAMERA_SCLK                                  1440000   // usually, Pixel ClockX2
```

```
…

/* notes :
After chaning preview mode to capture mode, some frames have to be skipped to provide stable image. FRAMESKIP_COUNT_FOR_CAPTURE defines how many frames will be skipped.
But this value has to be decided accroding to guide of module vendor.
*/
#define FRAMESKIP_COUNT_FOR_CAPTURE 1

/* notes :
Define image resolution from camera module during preview mode and capture mode.
And define total number of Zoom step and size per each step.
Following shows an example to device crop region in preview mode.
*/
// ZOOM Setting!!
#define PRV_W                   1280   // width, preview size according to Camera module
#define PRV_H                   960    // heigh
#define PRV_ZOFFX               40     // zoom width-gap in case of one step in preview
#define PRV_ZOFFY               30      // zoom height-gap

#define CAP_W                   2592   // width, preview size according to Camera module
#define CAP_H                   1944   // heigh
#define CAP_ZOFFX               80     // zoom gap in case of one step in Capture
#define CAP_ZOFFY               60     // zoom height-gap

/* notes :
Define minimum resolution required to change to capture mode from preview mode.
For example, capture mode is changed if capture size is bigger than 800.
*/
#define CAM_2XMAX_ZOOM_STEP         5    // Total Zoom Step
#define CAM_CAPCHG_WIDTH            1280

#ifdef CONFIG_USE_ISP
#define CAM_MAX_ZOOM_STEP       CAM_2XMAX_ZOOM_STEP+1
#endif
```

Table 14. Example of clock setting for camera sensor module

2) Fill out other features

In Case of add new camera module, you have to make needful feature to new camera module.
Please refer to the yellow shaded texts in Table15.

Android_home\kernel/driver/media/video/tcccam/sensor_if.h

```
/*
// set in case Sensor support zoom function. TCC zoom will be use if not define.
#define USE_SENSOR_ZOOM_IF
//set in case Sensor support effect function. TCC effect will be use if not define.
#define USE_SENSOR_EFFECT_IF
//set h/w i2c comm. Function. s/w i2c comm. will be use if not define.
#define USING_HW_I2C
//Sensor Resolution feature.
#define SENSOR_2M (SENSOR_5M / SENSOR_3M)
*/

#if defined(CONFIG_VIDEO_CAMERA_SENSOR_AIT848_ISP)
#define SENSOR_5M
#define USE_SENSOR_ZOOM_IF
```

```
#define USE_SENSOR_EFFECT_IF
#include "venus_ait848_5mp.h"
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9P111)
#define SENSOR_5M
#define USING_HW_I2C
#include "mt9p111_5mp.h"
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_MV9317)
#define SENSOR_3M
#define USING_HW_I2C
#include "mv9317_3mp.h"
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9D112)
#define SENSOR_2M
#define USING_HW_I2C
#include "mt9d112_2mp.h"
#elif defined(CONFIG_VIDEO_CAMERA_SENSOR_S5K4BAFB)
#define SENSOR_2M
#define USING_HW_I2C
#define TCC_VCORE_30FPS_CAMERASENSOR
#include "s5k4bafb_2mp.h"
#endif
```

Table 15. Example of Filling out other features

- 4.3.2 Add Kernel init information In Case of Dual camera

1) Fill out Sensor header file information.

The sensor_info_init_mt9p111 in Table16 is initialization information function of dual camera sensor. You have to modify the feature to set the frequency of clock in dual camera according to the architecture for the camera module. Please refer to the yellow shaded texts in Table16.

In TCC 89xx demo board, you have to modify the yellow shaded texts as below.
The Section 4.3.1, ‘Add Kernel init information In case of Single Camera’, explains the procedures for setting the frequency of PLL.

Refer to the green shaded text in Table16 for setting other parameters of dual camera

Android_home\kernel/driver/media/video/tcccam/mt9p111_5mp.c

```
#if defined(CONFIG_VIDEO_DUAL_CAMERA_SUPPORT)
void sensor_info_init_mt9p111(TCC_SENSOR_INFO_TYPE *sensor_info)
{
        sensor_info->i2c_addr                   = 0x7A;
        sensor_info->reg_term                   = 0x0000;   // Don’t Fix it!
        sensor_info->val_term                   = 0x0000;   // Don’t Fix it!

        #if defined(CONFIG_ARCH_TCC92XX)
        sensor_info->m_clock                    = 240000;
        sensor_info->s_clock                    = 1440000;

        #elif defined(CONFIG_ARCH_TCC93XX)
        #if defined(CONFIG_USE_ISP)
        sensor_info->m_clock                    = 240000;
        sensor_info->s_clock                    = 1680000;
        #else // CONFIG_USE_ISP
        sensor_info->m_clock                    = 240000;
        sensor_info->s_clock                    = 1680000;
        #endif// CONFIG_USE_ISP
```

```
        #elif defined(CONFIG_ARCH_TCC88XX)
        #if defined(CONFIG_USE_ISP)
        sensor_info->m_clock                    = 240000;
        sensor_info->s_clock                    = 1440000;
        #else   // CONFIG_USE_ISP
        sensor_info->m_clock                    = 240000;
        sensor_info->s_clock                    = 1440000;
        #endif   // CONFIG_USE_ISP
        #endif
        sensor_info->preview_w                  = 1280;
        sensor_info->preview_h                  = 960;
        sensor_info->preview_zoom_offset_x      = 16;
        sensor_info->preview_zoom_offset_y      = 12;
        sensor_info->capture_w                  = 2592;
        sensor_info->capture_h                  = 1944;
        sensor_info->capture_zoom_offset_x      = 32;
        sensor_info->capture_zoom_offset_y      = 24;
        sensor_info->max_zoom_step              = 24;
        sensor_info->cam_capchg_width           = 1280;
        sensor_info->p_clock_pol                = NEGATIVE_EDGE;
        sensor_info->v_sync_pol                 = ACT_HIGH;
        sensor_info->h_sync_pol                 = ACT_HIGH;
        #if defined(CONFIG_ARCH_TCC892X)
        sensor_info->de_pol                     = ACT_LOW;
        #endif
        sensor_info->format                     = M420_ZERO;
        sensor_info->capture_skip_frame         = 1;
        sensor_info->sensor_sizes               = sensor_sizes_mt9p111;
}
```

Table 16. Setting of clock and others information

2) Fill out Other Features

If it is dual camera, you have to check the information of camera for both of the two cameras.
And you have to make needful feature to new camera modules.
Please refer to Table15 and 4.3.1.-2)

3) Registration initialization information function

Please refer to functions in Table16 to registration of Initialization Information
The sensor_if_set_facing_front() in Table 17 call the sensor_info_init_mt9p111 in Table16.

- Functions to setting of Initialization information
sensor_if_set_facing_front()
sensor_if_set_facing_back()

- Functions to setting of initialization func
sensor_init_func_set_facing_front()
sensor_init_func_set_facing_back()

Android_home\kernel/driver/media/video/tcccam/sensor_if.c

```
#if defined(CONFIG_VIDEO_DUAL_CAMERA_SUPPORT)
void sensor_if_set_facing_front(void)
{
        #if defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9M113)
                sensor_info_init_mt9m113(&tcc_sensor_info);
```

```
        #endif
}

void sensor_if_set_facing_back(void)
{
        #if defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9P111)
                sensor_info_init_mt9p111(&tcc_sensor_info);
        #endif
}

void sensor_init_func_set_facing_front(SENSOR_FUNC_TYPE *sensor_func)
{
        #if defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9M113)
                sensor_init_fnc_mt9m113(sensor_func);
        #endif
}

void sensor_init_func_set_facing_back(SENSOR_FUNC_TYPE *sensor_func)
{
        #if defined(CONFIG_VIDEO_CAMERA_SENSOR_MT9P111)
                sensor_init_fnc_mt9p111(sensor_func);
        #endif
}
#endif
```

Table 17. Registration of Initialization Information

- 4.3.2 Add initialization information in HAL layer

In case of well known resolution, you won't have to modify the whole camera part of HAL layer to add new camera.
If there is unknown resolution in TelechipsCameraHardware::initDefaultParameters(), you must add parameter in the function and refer to Table18 for the function.
And sometimes the resolution of output in camera sensor is different from the setting resolution in HAL layer.
If you have properly setting of camera resolution in HAL layer, camera block finally output size of image as setting size in HAL layer.

** well known resolution : Android_home/packages/apps/camera/res/values/strings.xml 참조

Android_home\hardware\telechips\common\libcamera\src\TelechipsCameraHardware.cpp

```
void TelechipsCameraHardware::initDefaultParameters()
{
…
        p.setPreviewSize(640, 480);
        if(v4L2Camera->mZoomSupport)        p.set("zoom-supported", "true");
        else                                p.set("zoom-supported", "false");
        p.set("zoom", "0");
        p.set("zoom-ratios", "100,120,140,160,180,200,220,240,260,280,300,320,340,360,380,400");
        p.set("max-zoom", "15");
        p.set("smooth-zoom-supported", "false");
        mMaxSkipframe_cnt = 20;
        switch(v4L2Camera->sensor_max_resolution) {
            case QQXGA:     // 5m sensor
                    p.setPreviewSize(1280, 960);
                    p.setPictureSize(2560, 1920);
                    p.set("picture-size-values", "2560x1920,2048x1536,1600x1200,1024x768");
```

```
                p.set("zoom-ratios", "100,120,140,160,180,200");
                p.set("max-zoom", "5");
                break;
        case QXGA:              // 3m sensor
                p.setPictureSize(2048, 1536);
                p.set("picture-size-values", "2048x1536,1600x1200,1024x768");
                break;
        case UXGA:              // 2m sensor
                p.setPictureSize(1600, 1200);
                p.set("picture-size-values", "1600x1200,1024x768");
                break;
        case SXGA:              // 1.3m sensor
                p.setPictureSize(1280, 1024);
                p.set("picture-size-values", "1280x1024,1024x768");
                break;
        case XGA:               // analog tv
                p.setPreviewSize(720, 480);
                p.setPictureSize(1600, 1200);
                p.set("picture-size-values", "1600x1200,1024x768");
                break;
        case VGA:               // vga sensor
                p.setPictureSize(640, 480);
                p.set("picture-size-values", "640x480,352x288,320x240,176x144");
                break;
        }
…
}
```

Table 18. Adding of initialization information in HAL layer

4.4. Make Kconfig and Makefile

If you must add items in menucofig under the kernel, refer to two files of Kconfig and Makefile in Android_home/kernel/drivers/media/video/tcccam.

- 4.4.1 Make Kconfig

Table19 explains how to edit Kconfig if you must add items in menuconfig under the kernel and the path of Kconfig is Android_home/kernel/drivers/media/video/tcccam/.

```
# In case of Single camera

config VIDEO_CAMERA_SENSOR_MT9P111
        tristate "MT9P111 5MP-sensor support"
        depends on VIDEO_SINGLE_CAMERA_SUPPORT
        help
            camera sensor support for 5MP

# In case of Dual camera

config VIDEO_DUAL_BACK_CAMERA_SUPPORT
        tristate "Select Telechips Back-Camera"
        depends on VIDEO_DUAL_CAMERA_SUPPORT
        help
            tcc dual-camera suport

config VIDEO_CAMERA_SENSOR_MT9P111
        tristate "MT9P111 5MP-sensor support"
        depends on VIDEO_DUAL_BACK_CAMERA_SUPPORT
        help
            camera sensor support for 5MP

config VIDEO_DUAL_FRONT_CAMERA_SUPPORT
        tristate "Select Telechips Front-Camera"
        depends on VIDEO_DUAL_CAMERA_SUPPORT
        help
            tcc dual-camera suport

config VIDEO_CAMERA_SENSOR_MT9M113
        tristate "MT9M113 1.3MP-sensor support"
        depends on VIDEO_DUAL_FRONT_CAMERA_SUPPORT
        help
        camera sensor support for 1.3MP
```

Table19. Example of Mapping Kconfig in Camera

- 4.4.2 Make Makefile

Table20 explains how to edit Makefile if you must add items in menuconfig under the kernel and the path of Makefile is Android_home/kernel/drivers/media/video/tcccam/.

```
…
…
obj-$(CONFIG_VIDEO_CAMERA_SENSOR_MT9P111) += mt9p111_5mp.o
…
obj-$(CONFIG_VIDEO_CAMERA_SENSOR_MT9M113) += mt9m113_1.3mp.o
…
…
```

Table 190. Example of Mapping Makefile in Camera

4.5. Menuconfig Configuration

- 4.5.1 In case of Single camera

To use single camera, you must refer to Table21.
For example, select MT9D112 which is 2M pixel sensor in menuconfig.

```
<*>   Telechips TCCXXX Camera support (EXPERIMENTAL)
< >     Enable Camera with max-clock
<*>     CAMERA sensor support
< >       Enable Telechips Dual-Camera
<*>       Enable Telechips Single-Camera
<*>         MT9D112 2MP-sensor support
< >         OV3640 3MP-sensor support (NEW)
< >         S5K4BAFB 2MP-sensor support (NEW)
< >         MV9317 3MP-sensor support (NEW)
< >         MT9P111 5MP-sensor support
< >         AIT848_ISP 5MP-sensor support (NEW)
< >         ISX006 5MP-sensor support (NEW)
< >         GT2005 2MP-sensor support (NEW)
< >         OV7690 VGA-sensor support (NEW)
< >         NEXTCHIP_TEST support (NEW)
< >         MT9M113 1.3MP-sensor support
< >     ISP support
```

Table 201. Menuconfig setting for Single Camera

- 4.5.2 In case of Dual camera

To use dual camera, you must select two camera modules in menuconfig.

Device Drivers →
Multimedia support →
[*] Video capture adapters →
[*] V4L platform devices →
Refer to Table22 for next paths.

Currently, we had tested only the dual camera that was made configure with the MT9P111 and MT9M113 in tcc892x and tcc893x demo board.
The MT9P111 is 5M pixel Camera module and MT9M113 is 1.3M pixel.
2M sensor and VGA module is expected to test within next month.

```
<*>   Telechips TCCXXX Camera support (EXPERIMENTAL)
< >     Enable Camera with max-clock
<*>     CAMERA sensor support
<*>       Enable Telechips Dual-Camera
<*>         Select Telechips Back-Camera
<*>           MT9P111 5MP-sensor support
<*>         Select Telechips Front-Camera
<*>           MT9M113 1.3MP-sensor support
< >       Enable Telechips Single-Camera
< >     ISP support
```

Table 212. Menuconfig setting for Dual Camera

Also, for dual camera UI configuration, you have to change HAL layer like below Table23.
In usual, default setting is single camera UI. If you want to use single camera UI, you have to block out "DUAL_CAMERA_SUPPORT" feature.

File path : Android_home\hardware/telechips/common/libcamera/include/TCC_V4l2_Camera.h

```
…
typedef enum {
        QQXGA, QXGA, UXGA,    SXGA, XGA, SVGA, VGA, QVGA,   QCIF
} image_size;
//#define SENSOR_5M
//#define SENSOR_3M
//#define SENSOR_2M
//#define SENSOR_VGA
//#define SENSOR_TVP5150
//#define SENSOR_RDA5888

#define DUAL_CAMERA_SUPPORT    // defalult : single camera

…
```

Table 223. Setting of dual camera UI configuration

4.6. Adjust Camcording Resolution

The camcorder resolution of android Camera supports two kinds of the high and low.
Please refer to Table24.

- File Path
TCC892x : Android_home\device/telechips/tcc892x-common/media_profiles.xml
TCC893x : Android_home\device/telechips/tcc893x-common/media_profiles.xml

```
<CamcorderProfiles cameraId="0">    // For single camera and Back camera of Dual camera

        <EncoderProfile quality="high" fileFormat="3gp" duration="60">
            <Video codec="h264"
                    bitRate="8000000"
                    width="1280"
                    height="720"
                    frameRate="20" />

            <Audio codec="amrnb"
                    bitRate="48000"
                    sampleRate="8000"
                    channels="1" />
        </EncoderProfile>

        <EncoderProfile quality="low" fileFormat="3gp" duration="60">
            <Video codec="h264"
                    bitRate="192000"
                    width="176"
                    height="144"
                    frameRate="20" />

            <Audio codec="amrnb"
                    bitRate="48000"
                    sampleRate="8000"
                    channels="1" />

        </EncoderProfile>

        <ImageEncoding quality="90" />
        <ImageEncoding quality="80" />
        <ImageEncoding quality="70" />
        <ImageDecoding memCap="20000000" />

        <Camera previewFrameRate="0" />

    </CamcorderProfiles>

    <CamcorderProfiles cameraId="1">    // for Front camera of Dual camera

        <EncoderProfile quality="high" fileFormat="3gp" duration="60">
            <Video codec="h264"
                    bitRate="8000000"
                    width="640"
                    height="480"
                    frameRate="20" />
```

```
            <Audio codec="amrnb"
                    bitRate="48000"
                    sampleRate="8000"
                    channels="1" />
        </EncoderProfile>

        <EncoderProfile quality="low" fileFormat="3gp" duration="60">
            <Video codec="h264"
                    bitRate="192000"
                    width="176"
                    height="144"
                    frameRate="20" />

            <Audio codec="amrnb"
                    bitRate="48000"
                    sampleRate="8000"
                    channels="1" />

        </EncoderProfile>

        <ImageEncoding quality="90" />
        <ImageEncoding quality="80" />
        <ImageEncoding quality="70" />
        <ImageDecoding memCap="20000000" />

        <Camera previewFrameRate="0" />

    </CamcorderProfiles>
```

Table 24. Setting of Camcording Resolution

4.7. Distinguish DE signal and HS signal for TCC892x / TCC893x

4.7.1 HS signal and DE signal

When TCC892x / TCC893x is connected is camera module or TV decoder, you connect HS signal and DE Signal carefully. You moust connected the HS-to-HS, DE-to-DE.
Generally, the signal from the Camera module is DE signal.

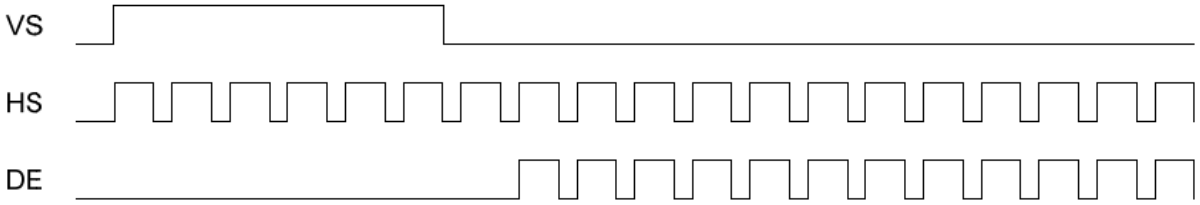


Table 25. Diffrence HS&DE signal

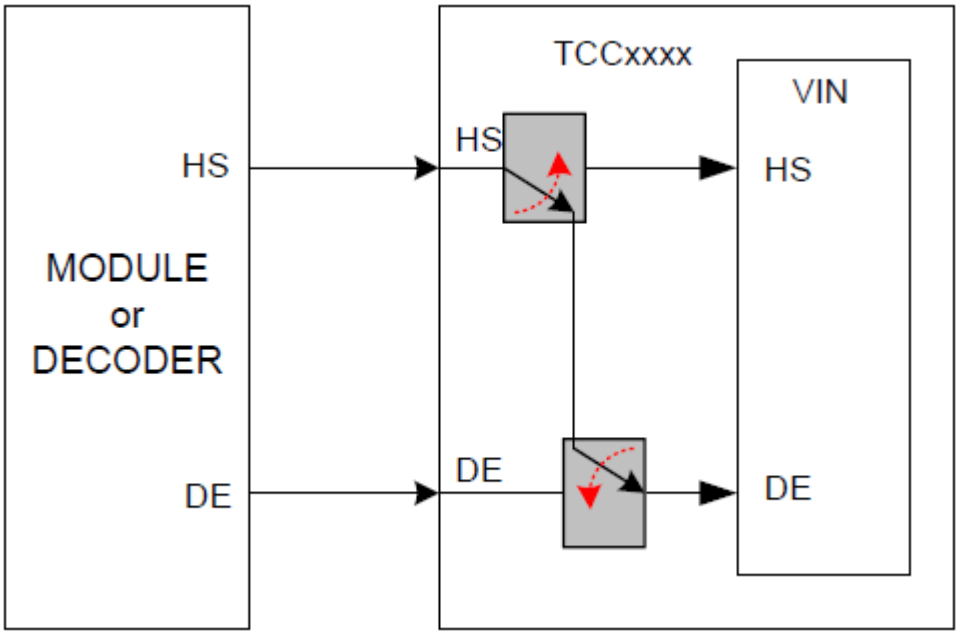


Table 236. The method of HS & DE connection if board connection is wrong

4.7.2 How to setting Video in block when use DE signal or HS signal

In case of M805-8923 board, Camera module DE signal connected to TCC8923 HS port, the other Case (M805-8925 or TCC89xx demo board), Camera module DE signal connected to TCC89xx DE port.
CAM_HS means DE signal

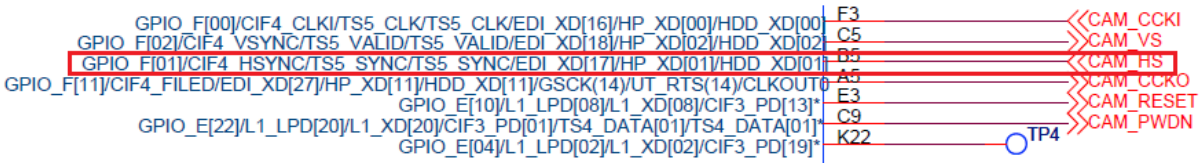


Table27. camera module DE signal connected to TCC89xx HS port

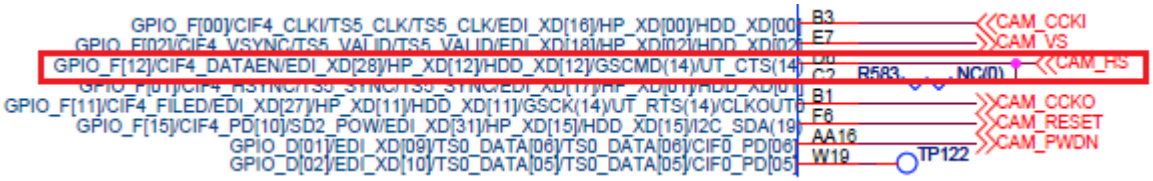


Table28. Camera module DE signal connected to TCC89xx DE port

If connected camera module DE signal connected to TCC89xx HS port, you should setting H-sync polarity, and the other case you should setting DE polarity as follows.

Android home/kernel/drivers/media/video/tcccam/tcc_cam.c

int tccxxx_vioc_vin_main(VIOC_VIN *pVIN) ()

```
#if defined(CONFIG_MACH_M805_892X) || defined(CONFIG_MACH_M805_893X)
  #if defined(CONFIG_M805S_8925_0XX) || defined(CONFIG_MACH_M805_893X)

    VIOC_VIN_SetSyncPolarity(pVIN, !(data->cif_cfg.polarity_href), !(data->cif_cfg.polarity_vsync),
                                OFF, data->cif_cfg.polarity_de, OFF, !(data->cif_cfg.polarity_pclk));
    VIOC_VIN_SetCtrl(pVIN, OFF, OFF, OFF, FMT_YUV422_8BIT, ORDER_RGB);

  #else // M805_8923 board. not use DE signal.
    VIOC_VIN_SetSyncPolarity(pVIN, !(data->cif_cfg.polarity_href), !(data->cif_cfg.polarity_vsync),
                                OFF, data->cif_cfg.polarity_de, OFF, !(data->cif_cfg.polarity_pclk));
    VIOC_VIN_SetCtrl(pVIN, OFF, ON, ON, FMT_YUV422_8BIT, ORDER_RGB);

#else

  VIOC_VIN_SetSyncPolarity(pVIN, !(data->cif_cfg.polarity_href), !(data->cif_cfg.polarity_vsync),
                              OFF, data->cif_cfg.polarity_de, OFF, !(data->cif_cfg.polarity_pclk));
  VIOC_VIN_SetCtrl(pVIN, OFF, OFF, OFF, FMT_YUV422_8BIT, ORDER_RGB);

#endif
```

5. FAQ

5.1. How to enable camera log through UART

To enable UART log for camera, please enable following features.

Android_home/kernel/drivers/media/video/tcccam/camera_core.c

```
@@ -61,7 +61,7 @@
 #include <linux/jiffies.h>
 extern unsigned long volatile __jiffy_data jiffies;

- #if 0
+# if 1
 static int debug       = 1;
 #else
 static int debug       = 0;
```

Android_home/kernel/drivers/media/video/tcccam/tcc_cam.c

```
@@ -60,7 +60,7 @@
 #include "tccisp/isp_interface.h"
 #endif

-#if 0
+#if 1
 static int debug       = 1;
 #else
 static int debug       = 0;
```

5.2. Required information to clear problems

1. UART Message(Camera log) Refer to FAQ 5.1.
2. DDMS Message log
3. Sensor information
   - Model name of sensor, Max resolution
   - Mclock , Pclock
   - Sensor initialization code, Sensor specification documentation

5.3. Setting functions to start and stop camera

Refer to Android_home/kernel/drivers/media/video/tcccam/mt9d112_2mp.c

```
1.Configure according to power-up sequence provided from module company.
2.Configure GPIO function according to H/W configuration for camera module.

static int sensor_open_mt9d112(void)
{
        sensor_power_disable();
        sensor_delay(20);

        sensor_power_enable();
        sensor_delay(20);
```

```
        sensor_powerdown_disable();
        sensor_delay(40)

        sensor_reset_low()
        sensor_delay(40)

        CIF_Open();
        sensor_delay(40);

        sensor_reset_high();
        sensor_delay(100);

        return write_regs_mt9d112(sensor_reg_common_mt9d112[0]);
}

static int sensor_close(void)
{
        CIF_ONOFF(OFF);             // TCC_CIF Block Off.

        sensor_powerdown_enable();
        msleep(60);

        sensor_reset_low();
        msleep(20);

        CIF_Close();    // Clock disable (MClk and TCC_CIF Scaler clk)
        msleep(5);

        return 0;
}
```

This function can be changed according to the actual sensor module.

5.4. Checking I2C and GPIO

1). Checking I2C signal by using oscilloscope
- Checking slave Address is correct
- Checking Initialization code is transmitted correctly.
- Checking there is ACK signal from sensor during Read/Write.

2). Checking I2C circuit

In case that 1 channel of the I2C has to control multiple devices, if there is camera device in the power off, other device on the i2c communication be disturbed intermittently.

To clear problem, you have to check Power-Down(Stand-by) mode in Camera Sensor specification documentation.
- If can use Power-Down mode :
  Use to Power-Down(Stand-by) sequence to Camera Stop.
  For example, mt9d112 can enable to Power-Down(Stand-by) mode.
  Refer to sensor_close_mt9d112() in mt9d112_2mp.c
- If there is no Power Down mode :
  This issue has to be checked by sensor module vendor.
  At the same time, check I2C sector on Telechips H/W guide document and inquire of Telechips H/W team about I2C circuit.

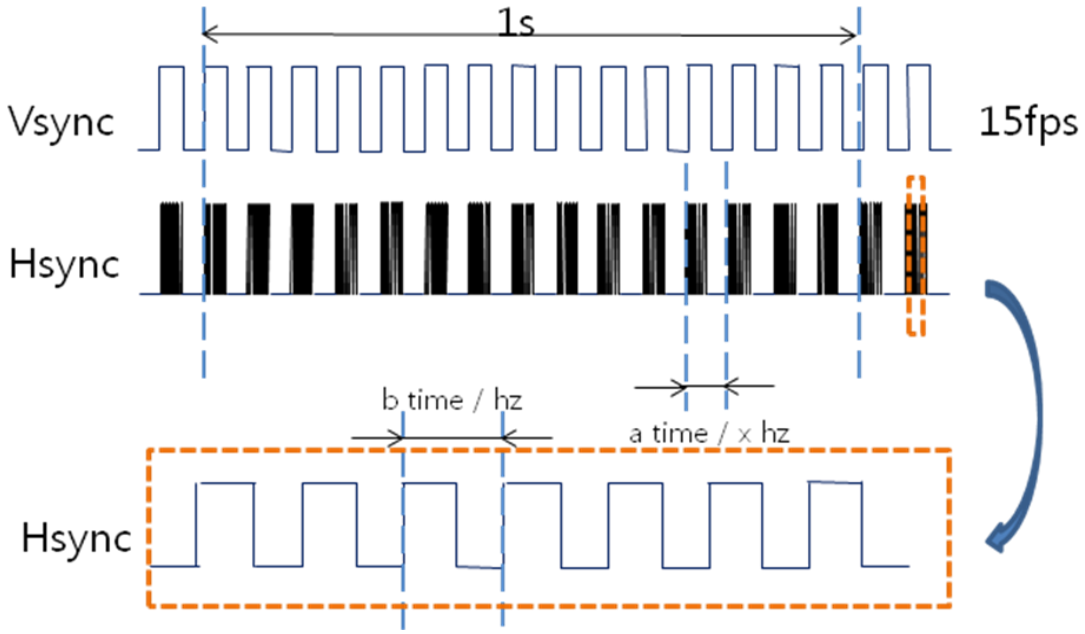
3). Checking I2C, GPIO setting firmware
- Checking Initialization code from sensor vendor is ported properly.
  Ex). In case of Mt9d112
    Checking array of sensor_initialize in mt9d112_2mp.c
    Checking Mt9d112_2mp.c wirte_regs() function step by step
- Compare GPIO setting with customer's schematic diagram
    Checking Camera sensor reset signal
      sensor_reset_high() & sensor_reset_low() in Sensor_if.c
    Checking camera sensor Power
      sensor_power_enable() & sensor_power_disaable()
      sensor_powerdown_enable() & sensor_powerdown_disaable() in Sensor_if.c

5.5. Recommendation zoom level for camera resolution

- VGA Camera sensor module:
  Do not use zoom because of image quality
- 1.3M ~ 3M pixel camera sensor module:
  Up to Zoom : 4x

5.6. Checking Vsync, Hsync Output

By checking vsync and hsync from sensor, you can check the sensor operates properly.



- Checking Vsync frequency
- Checking Hsync frequency.
  You can get Hsync frequency by dividing a by b.
  'a time = 66 ms
  'b time = 0.1375 ≒ 0.138
  66 ms / 0.138 ms ≒ (about) 480 hz
  It means number of horizontal line of sensor output is almost 480.
  And the output of sensor is correct if resolution is 640x480.

5.7. Issues during start camera function

- Checking each step using DDMS message
- Share log files with HQ

5.8. If there is no preview image when camera function is started

1. Checking interrupt is happened by enabling message of cif_cam_isr() in tcc_cam.c

kernel/drivers/media/video/tcccam/tcc_cam.c

```
cif_cam_isr()
{
……

if(TDD_CIF_GetIntStatus(GET_CIF_INT_ONEFRAME_STORE))
        {
                if(data->stream_state == STREAM_ON)
                {
                    dprintk("[Camera Preview] Interrupt Rising Up!!\n");
                    ……..
                }
         }
……..
}
```

2. If there is no interrupt, please check following items
   Power sequence (Refer to FAQ 5.3.)
   I2C status. (Refer to FAQ 5.4.)

3. If there is Interrupt, please check CIF/ISP block register
   In this case, there is a need to check it by HQ.
   Please send HQ log message.

5.9. If image quality is not good

This issue has to be checked by sensor module vendor.
Please contact to sensor module vendor.

5.10. Changing GPIO setting to control camera power

There are 3 GPIO signals to control power of mt9d112.
- CAM_PWR, CAM_PWDN, CAM_RESET

Customer has to modify following functions according to real hardware.

How to set GPIO as follows
Step 1. You have to initialize to the GPIO port which have direction for input or output.
The initialization of The GPIO port is done on sensor_if_init() in sensor_if.c.
Step 2. The GPIO port control is done on sensor_xxx_xxx() in sensor_if.c.
sensor_xxx_xxx() is sensor_reset_high(), sensor_reset_low(),
sensor_power_enable(), sensor_power_disable(),
sensor_powerdown_enable() and sensor_power_downdisable().

Source File : Sensor_if.c
Functions : sensor_if_init(),
sensor_reset_high() & sensor_reset_low()
sensor_power_enable() & sensor_power_disable()
sensor_powerdown_enable() & sensor_powerdown_disable()

In addition, those signals are assigned as follows in case of TCC89X Demo board.

CAM_PWR : Using signal from GPIO Expander IC

CAM_PWDN : GPIO_F_21 (Multiplexed with SD data0 of SD slot1)

CAM_RESET : GPIO_E_2

This assignment is only for demo board and should be changed according to the actual hardware - especially CAM_PWDN because it has alternative function- SDIO.
During camera initialization(TDD_CIF_Initialize() in tdd_cif.c), these ports are configured as GPIO.

If a customer uses this port for SDIO, SD Card or WiFi connected to SD slot1 will not work properly because its data 0 is configured as GPIO instead of SD data0.

5.11. Unstable screen during zooming, switching between camcorder and camera.

CIF scaler clock has to be adjusted according to output clock of camera sensor(Pixel clock) .
You have to find scaler clock to guarantee stable screen by decreasing scaler clock step by step.

- Stable range of CIF scaler clock : slower than 74MHz
  Ex) In case of Mt9d112 sensor module : (2M pixel) : 54Mhz
  In case of MV9317 sensor module : (3M pixel) : 72Mhz

- Recommended value for scaler clock is 2 x pixel clock.(P_clock)

If use ISP block, you can use more than 72Mhz clock. (=2 x P_clock)

5.12. How to change resolution and format

TCC89xx : device/telechips/tcc89xx-common/media_profiles.xml

1) Resolution

Max. and min resolution has to be chosen for the actual device
Ex) In case of 89xx
High : 1024x720
Low : 176x144

If changing camera encoding resolution, you have to change video bit rate.
Ex) In case of 640x480

- 1024x720 : 8000000 = 640x480 : X bit rate
- X bit rate = about 3300000 (bps)

Android_home/device/telechips/tcc89xx-common/media_profiles.xml

```
<EncoderProfile quality="high" fileFormat="3gp" duration="60">
    <Video codec="h264"
            bitRate = "8000000"   ➔  bitRate = "3300000
            width = "1024"        ➔  width = "640"
            height = "720"        ➔  height = "480"
            frameRate="20" />

    ……
</EncoderProfile>
```

2) Encoding Codec

- File format
  Éclair : 3GP/MP4/TS
  Froyo : 3GP/MP4
  Gingerbread : 3GP/MP4/TS
  JellyBean : 3GP/MP4/TS
  KitKat : 3GP/MP4/TS

- Video codec : M4V/H.264/H.263

- Audio codec : AAC/AMR NB

If changing encoding codec, See the following.

Android_home/device/telechips/tcc89XX-common/media_profiles.xml

```
<EncoderProfile quality="high" fileFormat="3gp" duration="60">
    <Video codec="h264"
            bitRate = "8000000"
            width = "1024"
            height = "720"
            frameRate="20" />

    <Audio codec="aac"
            bitRate = "48000"
            sampleRate = "8000"
            channels="1" />
</EncoderProfile>
```

5.13. How to obtain YUV data for JTAG debugger

If you want to obtain YUV data, you must use the JTAG debugger and do as follows

1) Find capture buffer address in cif_capture_dma_set() inside the tcc_cam.c

2) Break point setting in cif_cam_isr() inside the tcc_cam.c

```
static irqreturn_t cif_cam_isr_in8920(int irq, void *client_data /*, struct pt_regs *regs*/) {

    struct TCCxxxCIF *data = (struct TCCxxxCIF *)&hardware_data;
    struct tccxxx_cif_buffer *curr_buf, *next_buf;
    unsigned int next_num;

……

    if(data->cif_cfg.oper_mode == OPER_PREVIEW) { // preview operation.
        if(pWDMABase->uIRQSTS.nREG & VIOC_WDMA_IREQ_EOFF_MASK) {

……

            BITCSET(pWDMABase->uCTRL.nREG, 1<<16, 1<<16); // update WDMA
            BITCSET(pWDMABase->uIRQSTS.nREG, 1<<6, 1<<6); // clear EOFF
     }
     return IRQ_HANDLED;
```

```
    if(data->cif_cfg.oper_mode == OPER_CAPTURE) { // capture operation.
        if(pWDMABase->uIRQSTS.nREG & VIOC_WDMA_IREQ_EOFR_MASK) {

……

            BITCSET(pWDMABase->uCTRL.nREG, 1<<16, 1<<16); // update WDMA
            }
        }

        return IRQ_HANDLED;    -----> Hear is break pointer setting poing
    }
……

}
```

3) Dump YUV data.

5.14. The camera-supporting information according to demo-board

This list is the camera-supporting information according to demo-board.
Refer to follow table.

demoboard	CIF Block	ISP Interface	single camera	dual camera
Tcc 89xx	O	X	O	O