OV7740 Camera Module

Software Application Note

Last Modified: May. 10st, 2009

Document Revision: 1.0

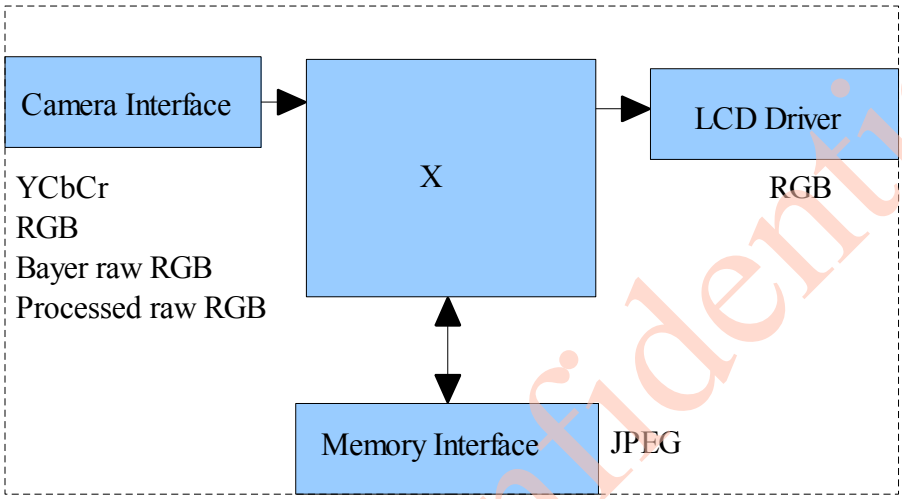
Sensor datasheet is the official document of OmniVision. Software/hardware/dual camera applicaton notes are application guide lines for reference. If there are any difference between sensor datasheet and application notes, please follow sensor datasheet and kindly report the difference to OVT FAE.

Table of Contents

1. Select Output format

OV7740 support output formats: RAW, RGB, YUV. How to choose the right output format for camera phone design or other applications? Let's look at the backend chip first.

The general diagram of backend chip is as below:



The data format at LCD driver are always RGB. For example, RGB444, RGB565, RGB555, RGB888 etc. The data format and memory interface are always Compression. The Compression data is compressed from YCbCr data. So Both RGB and YCbCr data are needed inside the backend chip. The "X" block is different for different backend chips.

1.1 Backend with full ISP

This kind of backend has full ISP. It takes raw RGB input, doing interpolation to generate RGB24 and doing translation to generate YCbCr. This kind of backend could take either Bayer raw RGB or processed raw RGB.

The advantage of processed raw RGB over Bayer raw RGB is the output data are processed. Sensor functions such as defect pixel correction, lens correction, gamma, de-noise, sharpness, BLC etc. could be applied. Since the life time of backend chip is longer than image sensor, sometimes backend chips could not fix defects of new sensors if taken Bayer raw RGB. But the defects of new sensors could be fixed in processed raw RGB output.

If backend take Bayer raw RGB format from sensor, all the image process operations such as defect pixel correction, lens correction, gamma, color matrix, de-noise, sharpness, etc should be done by backend except BLC. If backend take processed raw RGB format from sensor, the image process operations such as defect pixel correction, lens correction, gamma, de-noise, sharpness etc could be done either inside sensor or by backend chips. In other words, user could select the image

process operation be done by which side.

1.2 Backend with YCbCr ISP

This kind of backend has ISP, but could take only YCbCr format. The ISP could convert YCbCr to RGB format for LCD display and compress YCbCr for storage.

1.3 Backend without ISP

This kind of backend doesn't have ISP built-in. It can not convert from one format to another by hardware. Actually the format conversion is done by software.

Sensor output YCbCr. Backend convert YCbCr to RGB for display by software.

1.4 Equations to Convert from One Format to Another

RGB24 to YCbCr

Y = 0.31R + 0.59G + 0.11B

Cb = 0.563(B-Y) + 128 = –0.175R – 0.332G + 0.501B + 128

Cr = 0.713(R-Y) + 128 = 0.492R – 0.420G – 0.078B + 128

Y = ((79 * R + 151 * G + 28 * B) >> 8);

Cb = ((-44 * R - 85 * G + 128 * B) >> 8) + 128;

Cr = ((126 * R - 107 * G - 20 * B) >> 8) + 128;

YCbCr to RGB24

R = Y + 1.405(Cr – 128)

G = Y – 0.698(Cr – 128) – 0.336(Cb – 128)

B = Y + 1.776(Cb – 128)

R = Y + (359*(Cr – 128)) >> 8

G = Y – (179*(Cr – 128) + 86*(Cb – 128))>>8

B = Y + (454*(Cb – 128)) >> 8

2. Select Output Resolution

2.1 Backend with ISP

If Backend chip has built-in ISP (Full ISP or YCbCr ISP), the ISP could do image scale. So OV7740 outputs only VGA format. ISP scaled VGA image to other resolution that mobile device needed.

2.2 Backend without ISP

If backend chip doesn't have image scale capability, then the Image scaler of OV7740 must be used to scale output resolution exactly the LCD size. For example, if the LCD size is 176x220, then the Image scaler will scale the output size to 176x220.

3. Adjust frame rate

The recommended frame rates are 30fps and 15fps for 60Hz light environment, 25fps and 14.3fps for 50Hz light environment. The frame rate for night mode is lower, we'll discuss night mode later. Pclk can be adjusted by PLL and digital Doubler , here changing PLL (register 0x11[7:6]) setting is not recommended.

Reference settings for above frame rates are listed below.

3.1 Frame Rate Adjustment for 24Mhz input clock

60 fps, PCLK = 48Mhz

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0xf0);
write_SCCB(0x2c, 0x01);
```

30 fps, PCLK = 24Mhz

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0xf0);
write_SCCB(0x2c, 0x01);
```

25 fps, PCLK = 24Mhz

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x5e);
write_SCCB(0x2c, 0x02);
```

15fps, PCLK = 12Mhz

SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x03);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0xf0);
write_SCCB(0x2c, 0x01);

3.2 Frame Rate Adjustment for 26 Mhz input clock

60 fps, PCLK = 52Mhz

SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x23);
write_SCCB(0x2c, 0x02);

30 fps, PCLK = 26Mhz

SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x23);
write_SCCB(0x2c, 0x02);

25fps, PCLK = 26Mhz

SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x90);
write_SCCB(0x2c, 0x02);

15 fps, PCLK = 13Mhz

SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x03);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x23);
write_SCCB(0x2c, 0x02);

3.3 Frame rate adjustment for 13 Mhz input clock

30 fps, PCLK = 26Mhz

SCCB_salve_Address = 0x42;

write_SCCB(0x11, 0x00);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x23);
write_SCCB(0x2c, 0x02);

25fps, PCLK = 26Mhz

SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x90);
write_SCCB(0x2c, 0x02);

15 fps, PCLK = 13Mhz

SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x23);
write_SCCB(0x2c, 0x02);

4. Night Mode

There are 2 types of settings for night mode. One type is set to fixed low frame rate, for example 3.75fps. The other type is set to auto frame rate, for example from 30fps to 3.75fps. When environment is bright, the frame rate is increased to 30fps. When environment is dark, the frame rate is decreased to 3.65fps.

4.1 Night Mode with Fixed Frame Rate

For 24Mhz Clock Input

3.75fps night mode for 60Hz light environment
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x03);
write_SCCB(0x55, 0x00);
write_SCCB(0x15, 0x00);

4.2 Night Mode with Auto Frame Rate

For 24Mhz Clock Input

30fps ~ 3.75fps night mode for 60Hz light environment

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0xf0);
write_SCCB(0x2c, 0x01);
write_SCCB(0x15, 0xF4);
```

15fps ~ 3.75fps night mode for 60Hz light environment

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x03);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0xf0);
write_SCCB(0x2c, 0x01);
write_SCCB(0x15, 0xB4);
```

25fps ~ 3.125fps night mode for 50Hz light environment

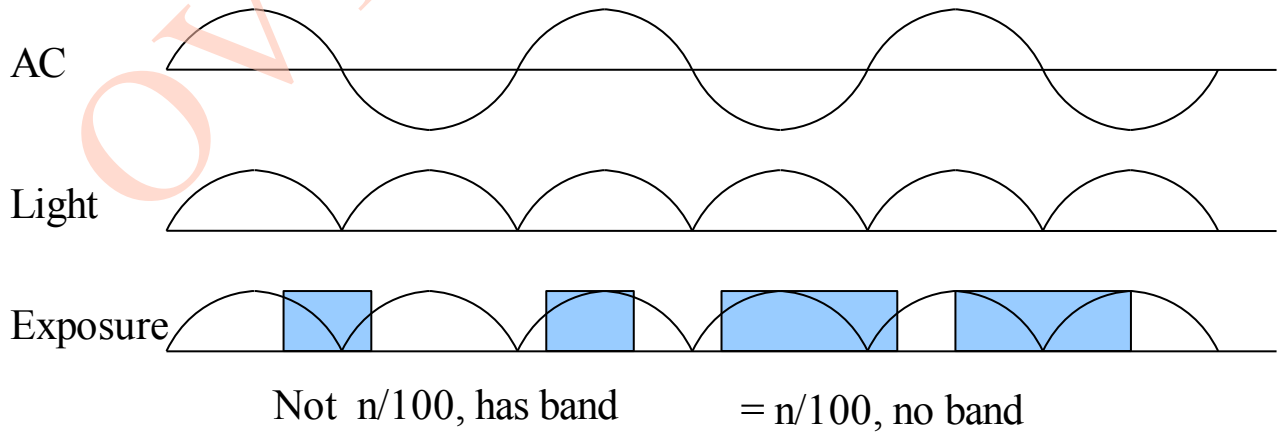
```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x55, 0x40);
write_SCCB(0x2b, 0x5e);
write_SCCB(0x2c, 0x02);
write_SCCB(0x15, 0xF4);
```

Note:

When OV7740 is set to low frame rate, there may be many white pixels shown on LCD of mobile phone or on PC diaplay.

5. Remove Light Band

5.1 Light Band



The strength of office light is not even. It changes with AC frequency. For example, if the AC frequency is 50Hz, the light changes strength at 100hz.

5.2 Remove Light band

Light band is removed by set exposure to n/100 (n/120 for 60Hz)seconds. The banding filter value tell OV7740 how many lines is 1/100 (1/120 for 60Hz) seconds.

5.3 Select Banding Filter by Region Information

The region information of mobile phone could be used to select banding filter values. A light frequency table is built to indicate which region uses 50Hz light and which region uses 60Hz light. When region information is got, the light frequency information could be get from the table.

Different frame rate could be used for different light frequency. So the frame rate is optimized for both 50hz light condition and 60hz light condition.
!

Banding Filter Setting for 24Mhz Input Clock

30fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x97); //50Hz banding filter
write_SCCB(0x51, 0x7e); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x23); //banding setp

15fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x4b); //50Hz banding filter
write_SCCB(0x51, 0x3f); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x57); //banding setp

25fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x97); //50Hz banding filter
write_SCCB(0x51, 0x7e); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x23); //banding setp

14.3fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x4b); //50Hz banding filter
write_SCCB(0x51, 0x3f); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x57); //banding setp

Banding Filter Setting for 13Mhz/26Mhz Input Clock

30fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0xa4); //50Hz banding filter
write_SCCB(0x51, 0x88); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x23); //banding setp

15fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x52); //50Hz banding filter
write_SCCB(0x51, 0x44); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x57); //banding setp

25fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0xa4); //50Hz banding filter
write_SCCB(0x51, 0x88); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x23); //banding setp

14.3fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x52); //50Hz banding filter
write_SCCB(0x51, 0x44); //60Hz banding filter

write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x57); //banding setp

5.4 Select Banding Filter by Automatic Light Frequency Detection

Set same frame rate for 50Hz and 60Hz light environment, set 50Hz and 60Hz banding filter value. OV7740 could automatic select 50Hz or 60Hz banding filter based on light frequency detection, the setting of automatic light frequency detection is dependent on the input clock frequency. Please contact OmniVision FAEs for the setting.
!

Banding Filter Setting for 24Mhz Input Clock

30fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x97); //50Hz banding filter
write_SCCB(0x51, 0x7e); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x23); //banding setp

15fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x4b); //50Hz banding filter
write_SCCB(0x51, 0x3f); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x57); //banding setp

Banding Filter Setting for 13Mhz/26Mhz Input Clock

30fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0xa4); //50Hz banding filter
write_SCCB(0x51, 0x88); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x23); //banding setp

15fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //banding filter enable
write_SCCB(0x50, 0x52); //50Hz banding filter
write_SCCB(0x51, 0x44); //60Hz banding filter
write_SCCB(0x52, 0x00);
write_SCCB(0x21, 0x57); //banding setp

5.5 When Light Band can not be Removed

Normally the light band is removed by banding filter.

But there is some special conditions such as mix light of sun light and office light, take picture of florescent light, the light band can not removed. The reason is the exposure time is less than 1/100 second for 50hz light environment and less than 1/120 second for 60hz light environment, so the light band can not be removed.

The light band is this conditions could not be removed for all CMOS sensors, not only OV7740. So there is no way to remove light band in this condition.

6. White Balance

OV7740 support simple white balance and advanced balance.

6.1 Simple White Balance

Simple white balance assume “gray world”. Which means the average color of world is gray. It is true for most environment.

Advantage of simple AWB

Simple white balance is not depend on lens. A general setting for simple white balance could applied for all modules with different lens.

Disadvantage of simple AWB

The color is not accurate in conditions where “gray world” not true. For example the background has a huge red, blue or green etc. the color of the foreground is not accurate. If the camera target single color such as red, blue, green, the simple white balance will make the single color gray.

```
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //AWB on
write_SCCB(0xAC, 0x6e); // Simple AWB
```

6.2 Advanced White Balance

Advanced white balance uses color temperature information to detect white area and do the white balance.

Advantage of Advanced AWB

Color is more accurate than simple white balance. Even the background is single color, the camera will not make the single color gray.

Disadvantage of Advanced AWB

Advanced white balance setting is depend on lens. The setting must be adjusted for every new type of lens. The adjustment must be done by OmniVision FAE in optical lab with some optical

equipment such as light box, color checker etc.

Settings

Contact with OmniVision local FAE.

6.3 How to select?

Generally, for low resolution camera module such as CIF, VGA and 1.3M, simple AWB is selected. For high resolution camera module such as 2M, 3M, advanced AWB is selected.

7. Defect Pixel Correction

Defect pixel include dead pixel and wounded pixel.

Dead pixel include white dead pixel and black dead pixel. White dead pixel is always white no matter the actual picture is bright or dark. Black dead pixel is always black no matter the actual picture is bright or dark.

Wounded pixel may change with light, but not as much as normal pixel. White wounded pixels are much brighter then normal pixels, but not complete white. Black wounded pixels are much darker than normal pixels, but not complete black.

OV7740 has built-in defect pixel correction function. If OV7740 output YCbCr, Processed raw RGB, the defect pixel correction function could be enabled to fix defect pixels. But if Bayer raw RGB is used, the defect pixel correction function of sensor could not be used. The defect pixel correction of backend chip should be used instead.

Please pay attention to the defect pixel correction function of backend chip. Some backend chip may not be able to correct all defect pixels of OV7740.

8. BLC

The function of Black Level Calibration (BLC) is to product accurate color in the dark area of picture. There is automatic BLC function built-in OV7740. It should always be turned on.

9. Video Mode

Video mode need high frame rate, usually fixed 15fps. There is no night mode for video mode.

10. Digital zoom

If OV7740 output image smaller than QVGA, it may support digital zoom. For example

VGA	not digital zoom supported
QVGA	1x, 2x
QQVGA	1x, 2x, 4x

QCIF	1x, 1.8x
QQCIF	1x, 2x, 3.6x

If backend chip support scale up, then more zoom level could be supported.

11. OV7740 Functions

11.1 Light Mode

Auto
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //AWB on
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);

Sunny
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //AWB off
write_SCCB(0x01, 0x5a);
write_SCCB(0x02, 0x5c);
write_SCCB(0x03, 0x42);
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);

Cloudy
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //AWB off
write_SCCB(0x01, 0x58);
write_SCCB(0x02, 0x60);
write_SCCB(0x03, 0x40);
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);

Office
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //AWB off
write_SCCB(0x01, 0x84);
write_SCCB(0x02, 0x4c);
write_SCCB(0x03, 0x40);
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);

```
write_SCCB(0x2e, 0x00);

Home
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //AWB off
write_SCCB(0x01, 0x96);
write_SCCB(0x02, 0x40);
write_SCCB(0x03, 0x4a);
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);

Night
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xff); //AWB on
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);
write_SCCB(0x15, 0xf4);
```

11.2 Color Saturation

The color saturation of OV7740 could be adjusted. High color saturation would make the picture looks more vivid, but the side effect is the bigger noise and not accurate skin color.

```
Saturation + 4
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x80);
write_SCCB(0xDE, 0x80);

Saturation + 3
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x70);
write_SCCB(0xDE, 0x70);
```

Saturation + 2
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x60);
write_SCCB(0xDE, 0x60);

Saturation + 1
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x50);
write_SCCB(0xDE, 0x50);

Saturation 0
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x40);
write_SCCB(0xDE, 0x40);

Saturation -1
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x30);
write_SCCB(0xDE, 0x30);

Saturation - 2
SCCB_salve_Address = 0x42;

```
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x20);
write_SCCB(0xDE, 0x20);
```

Saturation - 3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x10);
write_SCCB(0xDE, 0x10);
```

Saturation - 4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x02;
write_SCCB(0xDA, temp);
write_SCCB(0xDD, 0x00);
write_SCCB(0xDE, 0x00);
```

11.3 Brightness

The brightness of OV7740 could be adjusted. Higher brightness will make the picture more bright. The side effect of higher brightness is the picture looks frogy.

Brightness +4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x0E);
write_SCCB(0xE3, 0x40);
```

Brightness +3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x0E);
write_SCCB(0xE3, 0x30);

Brightness +2
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x0E);
write_SCCB(0xE3, 0x20);


Brightness +1
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x0E);
write_SCCB(0xE3, 0x10);


Brightness 0
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x0E);
write_SCCB(0xE3, 0x00);

Brightness -1
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
```

```
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x06);
write_SCCB(0xE3, 0x10);
```

Brightness -2

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x06);
write_SCCB(0xE3, 0x20);
```

Brightness -3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x06);
write_SCCB(0xE3, 0x30);
```

Brightness -4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE4, 0x06);
write_SCCB(0xE3, 0x40);
```

11.4 Contrast

The contrast of OV7740 could be adjusted. Higher contrast will make the picture sharp. But the side effect is loosing dynamic range.

Contrast +4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x30);
write_SCCB(0xE3, 0x00);
temp = read_SCCB(0xE4);
temp &= 0xfb;
write_SCCB(0xE4, temp);
```

Contrast +3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x2c);
write_SCCB(0xE3, 0x00);
temp = read_SCCB(0xE4);
temp &= 0xfb;
write_SCCB(0xE4, temp);
```

Contrast +2

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x28);
write_SCCB(0xE3, 0x00);
temp = read_SCCB(0xE4);
temp &= 0xfb;
write_SCCB(0xE4, temp);
```

Contrast +1

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
```

```
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x24);
write_SCCB(0xE3, 0x00);
temp = read_SCCB(0xE4);
temp &= 0xfb;
write_SCCB(0xE4, temp);
```

Contrast 0

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x20);
write_SCCB(0xE3, 0x00);
temp = read_SCCB(0xE4);
temp &= 0xfb;
write_SCCB(0xE4, temp);
```

Contrast -1

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x1c);
write_SCCB(0xE3, 0x20);
temp = read_SCCB(0xE4);
temp |= 0x04;
write_SCCB(0xE4, temp);
```

Contrast -2

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
```

```
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x18);
write_SCCB(0xE3, 0x48);
temp = read_SCCB(0xE4);
temp |= 0x04;
write_SCCB(0xE4, temp);
```

Contrast -3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x14);
write_SCCB(0xE3, 0x80);
temp = read_SCCB(0xE4);
temp |= 0x04;
write_SCCB(0xE4, temp);
```

Contrast -4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x04;
write_SCCB(0xDA, temp);
write_SCCB(0xE1, 0x20);
write_SCCB(0xE2, 0x10);
write_SCCB(0xE3, 0xD0);
temp = read_SCCB(0xE4);
temp |= 0x04;
write_SCCB(0xE4, temp);
```

11.5 Special effects

OV7740 support some special effects such as B/W, negative, sepia, bluish, redish, greenish etc. If users need other special effects, it should be supported by backend chips.

Normal

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
```

```
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp &= 0xe7;
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0x80);
write_SCCB(0xE0, 0x80);
```

Antique

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x18;
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0x40);
write_SCCB(0xE0, 0xa0);
```

Bluish

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x18;
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0xa0);
write_SCCB(0xE0, 0x40);
```

Greenish

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x18;
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0x60);
write_SCCB(0xE0, 0x60);
```

Redish

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x18;
```

```
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0x80);
write_SCCB(0xE0, 0xc0);

B&W
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x18;
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0x80);
write_SCCB(0xE0, 0x80);

Negative
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp &= 0xe7;
temp |= 0x40;
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0x80);
write_SCCB(0xE0, 0x80);

B&W negative
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
temp = read_SCCB(0xDA);
temp |= 0x18;
temp |= 0x40;
write_SCCB(0xDA, temp);
write_SCCB(0xDF, 0x80);
write_SCCB(0xE0, 0x80);
```

11.6 YUV Sequence

```
SCCB_salve_Address = 0x42;
YUYV
temp = read_SCCB(0x0C);          //set 0x0c.bit4='0'
temp &= 0xef;
write_SCCB(0x0C, temp);
temp = read_SCCB(0xD9);          //set 0xd9.bit1='0'
temp &= 0xfd;
```

```
write_SCCB(0xD9, temp);
```

YVYU

```
temp = read_SCCB(0x0C);          //set 0x0c.bit4='0'
temp &= 0xef;
write_SCCB(0x0C, temp);
temp = read_SCCB(0xD9);          //set 0xd9.bit1='1'
temp |= 0x02;
write_SCCB(0xD9, temp);
```

UYVY

```
temp = read_SCCB(0x0C);          //set 0x0c.bit4='1'
temp |= 0x10;
write_SCCB(0x0C, temp);
temp = read_SCCB(0xD9);          //set 0xd9.bit1='0'
temp &= 0xfd;
write_SCCB(0xD9, temp);
```

VYUY

```
temp = read_SCCB(0x0C);          //set 0x0c.bit4='1'
temp |= 0x10;
write_SCCB(0x0C, temp);
temp = read_SCCB(0xD9);          //set 0xd9.bit1='1'
temp |= 0x02;
write_SCCB(0xD9, temp);
```

12. Deal with Lens

12.1 Light fall off

Light fall off means the corner of image is darker than center of image. It is caused by the lens. The lens shading correction function of OV7740 could be turned on to compensate the corner brightness and make the whole picture looks same bright.

12.2 Dark corner

Some lens may have dark corner. Dark corner means the color of picture looks almost black. It is not possible to correct dark corner with lens correction. So the module with dark corner is NG, it can not be used.

12.3 Resolution

The resolution of camera module depends on lens design, focus adjustment and sensor resolution as well. The focus adjustment is very important for camera module assembly.

For OV7740 the focus distance is about 40~50cm. The depth of field is about from 20~25cm to infinite. If checking resolution of camera module, the resolution chart should be placed 40~50 cm

away.

12.4 Optical contrast

The optical contrast of lens is very important to picture quality. If the optical contrast of lens is not good, the picture would looks forgy. Though it could be improved by increase the sensor contrast to make the picture sharper, the higher sensor contrast would make the detail lost of dark area of the picture.

12.5 Lens Cover

The lens cover is the cheapest part in optical path. But it could affect picture quality very much. The lens cover should be made with optical glass with AR coating at both side. Otherwise, the lens cover may cause sensitivity loss and/or stronger lens flare.

12.6 Lens Correction

Lens Correction setting should be tunned with every module. Please contact with OmniVision local FAE foe lens correction tuning.

13. Reference Settings

13.1 VGA YcbCr Preview

```
/************************
Input clock 24Mhz
VGA  YUV  30fps
************************/
write_SCCB(0x12, 0x80);
//delay 5ms
write_SCCB(0x13, 0x00);
write_SCCB(0x11, 0x01);
write_SCCB(0x12, 0x00);
write_SCCB(0xd5, 0x10);
write_SCCB(0x0c, 0x12);
write_SCCB(0x0d, 0x34);
write_SCCB(0x17, 0x25);
write_SCCB(0x18, 0xa0);
write_SCCB(0x19, 0x03);
write_SCCB(0x1a, 0xf0);
write_SCCB(0x1b, 0x89);
write_SCCB(0x22, 0x03);
write_SCCB(0x29, 0x17);
write_SCCB(0x2b, 0xf8);
write_SCCB(0x2c, 0x01);
write_SCCB(0x31, 0xa0);
write_SCCB(0x32, 0xf0);
write_SCCB(0x33, 0xc4);
```

```
write_SCCB(0x35, 0x05);
write_SCCB(0x36, 0x3f);
write_SCCB(0x04, 0x60);
write_SCCB(0x27, 0x80);
write_SCCB(0x3d, 0x0f);
write_SCCB(0x3e, 0x81);
write_SCCB(0x3f, 0x40);
write_SCCB(0x40, 0x7f);
write_SCCB(0x41, 0x6a);
write_SCCB(0x42, 0x29);
write_SCCB(0x44, 0xe5);
write_SCCB(0x45, 0x41);
write_SCCB(0x47, 0x42);
write_SCCB(0x48, 0x00);
write_SCCB(0x49, 0x61);
write_SCCB(0x4a, 0xa1);
write_SCCB(0x4b, 0x5e);
write_SCCB(0x4c, 0x18);
write_SCCB(0x4d, 0x50);
write_SCCB(0x4e, 0x13);
write_SCCB(0x64, 0x00);
write_SCCB(0x67, 0x88);
write_SCCB(0x68, 0x1a);
write_SCCB(0x14, 0x38);
write_SCCB(0x24, 0x3c);
write_SCCB(0x25, 0x30);
write_SCCB(0x26, 0x72);
write_SCCB(0x50, 0x97);
write_SCCB(0x51, 0x7e);
write_SCCB(0x52, 0x00);
write_SCCB(0x53, 0x00);
write_SCCB(0x20, 0x00);
write_SCCB(0x21, 0x23);
write_SCCB(0x38, 0x14);
write_SCCB(0xe9, 0x00);
write_SCCB(0x56, 0x55);
write_SCCB(0x57, 0xff);
write_SCCB(0x58, 0xff);
write_SCCB(0x59, 0xff);
write_SCCB(0x5f, 0x04);
write_SCCB(0xec, 0x00);
write_SCCB(0x13, 0xff);
write_SCCB(0x80, 0x7d);
write_SCCB(0x81, 0x3f);
write_SCCB(0x82, 0x32);
write_SCCB(0x83, 0x01);
write_SCCB(0x38, 0x11);
write_SCCB(0x84, 0x70);
```

```
write_SCCB(0x85, 0x00);
write_SCCB(0x86, 0x03);
write_SCCB(0x87, 0x01);
write_SCCB(0x88, 0x05);
write_SCCB(0x89, 0x30);
write_SCCB(0x8d, 0x30);
write_SCCB(0x8f, 0x85);
write_SCCB(0x93, 0x30);
write_SCCB(0x95, 0x85);
write_SCCB(0x99, 0x30);
write_SCCB(0x9b, 0x85);
write_SCCB(0x9c, 0x08);
write_SCCB(0x9d, 0x12);
write_SCCB(0x9e, 0x23);
write_SCCB(0x9f, 0x45);
write_SCCB(0xa0, 0x55);
write_SCCB(0xa1, 0x64);
write_SCCB(0xa2, 0x72);
write_SCCB(0xa3, 0x7f);
write_SCCB(0xa4, 0x8b);
write_SCCB(0xa5, 0x95);
write_SCCB(0xa6, 0xa7);
write_SCCB(0xa7, 0xb5);
write_SCCB(0xa8, 0xcb);
write_SCCB(0xa9, 0xdd);
write_SCCB(0xaa, 0xec);
write_SCCB(0xab, 0x1a);
write_SCCB(0xce, 0x78);
write_SCCB(0xcf, 0x6e);
write_SCCB(0xd0, 0x0a);
write_SCCB(0xd1, 0x0c);
write_SCCB(0xd2, 0x84);
write_SCCB(0xd3, 0x90);
write_SCCB(0xd4, 0x1e);
write_SCCB(0x5a, 0x24);
write_SCCB(0x5b, 0x1f);
write_SCCB(0x5c, 0x88);
write_SCCB(0x5d, 0x60);
write_SCCB(0xac, 0x6e);
write_SCCB(0xbe, 0xff);
write_SCCB(0xbf, 0x00);
write_SCCB(0x70, 0x00);
write_SCCB(0x71, 0x34);
write_SCCB(0x74, 0x28);
write_SCCB(0x75, 0x98);
write_SCCB(0x76, 0x00);
write_SCCB(0x77, 0x08);
write_SCCB(0x78, 0x01);
```

write_SCCB(0x79, 0xc2);
write_SCCB(0x7d, 0x02);
write_SCCB(0x7a, 0x4e);
write_SCCB(0x7b, 0x1f);
write_SCCB(0xec, 0x00);
write_SCCB(0x7c, 0x0c);

13.2 Output Format

YUV :
write_SCCB(0x12, 0x00);
write_SCCB(0x36, 0x3f);

RAW-10
write_SCCB(0x12, 0x01);
write_SCCB(0x36, 0x2f);
write_SCCB(0x83, 0x01);

RAW-8
write_SCCB(0x12, 0x01);
write_SCCB(0x36, 0x2f);
write_SCCB(0x83, 0x05);

Sensor RAW:
write_SCCB(0x12, 0x11);
write_SCCB(0x36, 0x3f);

13.3 Size

VGA 640x480:
write_SCCB(0x31, 0xa0);
write_SCCB(0x32, 0xf0);
write_SCCB(0x82, 0x32);

QVGA 320x240:
write_SCCB(0x31, 0x50);
write_SCCB(0x32, 0x78);
write_SCCB(0x82, 0x3f);

QQVGA 160x120:
write_SCCB(0x31, 0x28);
write_SCCB(0x32, 0x3c);
write_SCCB(0x82, 0x3f);

CIF 352x288:
write_SCCB(0x31, 0x58);
write_SCCB(0x32, 0x90);
write_SCCB(0x82, 0x3f);

Revision History

Rev1.00

.....