OV2640 Camera Module

Software Application Notes

Last Modified: Jan 11th, 2008

Document Revision: 1.04

Table of Contents

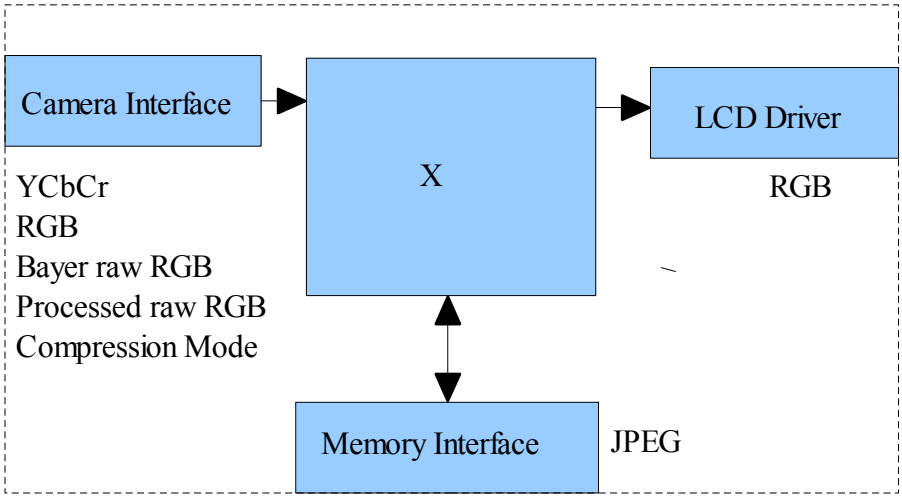
OV2640 Camera Module..... 1
Software Application Notes..... 1
1. How to Select Output format?..... 4
1.1 Back-end with full ISP..... 4
1.2 Back-end with YCbCr ISP..... 5
1.3 Back-end without ISP..... 5
1.4 Equations to Convert from One Format to Another..... 5
2. How to Select Output Resolution?..... 6
2.1 back-end with ISP..... 6
2.2 back-end without ISP..... 6
3. How to Adjust frame rate..... 6
3.1 SVGA Preview, 30fps, 24Mhz input clock..... 6
3.2 SVGA Preview, 15fps, 24 Mhz input clock..... 7
3.3 SVGA Preview, 25fps, 24 Mhz input clock..... 7
3.4 SVGA Preview, 14.3fps, 24 Mhz input clock..... 7
3.5 UXGA Capture, 7.5fps, 24 Mhz input clock..... 7
3.6 UXGA Capture, 7.14fps, 24 Mhz input clock..... 8
4. How to set Night Mode Preview..... 8
4.1 Night Mode with Fixed Frame Rate..... 8
4.2 Night Mode with Auto Frame Rate..... 8
5. How to Remove Light Band in Preview Mode..... 9
5.1 Light Band..... 9
5.2 Remove Light band..... 10
5.3 Select Banding Filter by Region Information..... 10
5.4 Select Banding Filter by Automatic Light Frequency Detection..... 11
5.5 Remove Light Band in Capture..... 11
5.6 When Light Band can not be Removed..... 12
6. White Balance..... 12
6.1 Simple White Balance..... 12
6.2 Advanced White Balance..... 12
6.3 How to select?..... 13
7. Defect Pixel Correction..... 13
8. BLC..... 13
9. Video Mode..... 13
10. Digital zoom..... 14
11. OV2640 Functions..... 14
11.1 Light Mode..... 14
11.2 Color Saturation..... 15
11.3 Brightness..... 16
11.4 Contrast..... 17
11.5 Special effects..... 18
11.6 YUV Sequence..... 20
12. Deal with Lens..... 21
12.1 Light fall off..... 21
12.2 Dark corner..... 21
12.3 Resolution..... 21

12.4 Optical contrast..... 21
12.5 Lens Cover..... 21
13. Reference Settings..... 22
13.1 YCbCr Reference Setting..... 22
13.1.1 SVGA Preview..... 22
13.1.2 UXGA Capture..... 26
13.2 RGB 565 Reference Setting..... 30
13.3 RGB raw Reference Setting..... 34
14. Capture Sequence..... 34
14.1 Shutter..... 35
14. 2 Dummy Lines..... 35
14.2.1 Extra Line..... 35
14.2.2 Dummy Line..... 35
14. 3 Dummy Pixels..... 36
14.4 Gain..... 36
14.5 Banding Filter..... 37
14.5.1 Preview..... 37
14.5.2 Capture..... 37
14.6 Auto frame rate..... 37
14.7 Capture Sequence..... 37
14.7.1 Preview..... 37
14.7.2 Stop Preview..... 38
14.7.3 Calculate Capture Exposure..... 38
14.7.4 Switch to UXGA..... 40
14.7.5 Write Registers..... 40
14.7.6 Capture..... 41
14.7.7 Back to preview..... 41

1. How to Select Output format?

OV2640 support 4 output format: YcbCr422,YCbCr420, RGB565, Bayer raw RGB and GRB, YUV422 JPEG. How to choose the right output format for camera phone design or other applications? Let's look at the back-end chip first.

The general diagram of back-end 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 JPEG. The JPEG data is compressed from YCbCr data. So Both RGB and YCbCr data are needed inside the back-end chip. The "X" block is different for different back-end chips.

1.1 Back-end with full ISP

This kind of back-end has full ISP. It takes raw RGB input, doing interpolation to generate RGB24 and doing translation to generate YCbCr. This kind of back-end could take 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, color matrix, de-noise, sharpness, BLC etc. could be applied. Since the life time of back-end chip is longer than image sensor, sometimes back-end 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 back-end 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, BCL etc should be done by back-end. If back-end take processed raw RGB format from sensor, the image process operations such as defect pixel correction, lens correction, gamma, color matrix, de-noise,

sharpness, BCL etc could be done either inside sensor or by back-end chips. In other words, user could select the image process operation be done by which side.

1.2 Back-end with YCbCr ISP

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

1.3 Back-end without ISP

This kind of back-end 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. There are 3 possible solution for this kind of back-end chips.

a. Sensor output YCbCr. back-end chip convert YCbCr to RGB for display by software.
b. Sensor output RGB565. Back-end chip convert RGB565 to YCbCR for JPEG compression.
c. Sensor output RGB565 for preview, output YCbCr for capture (JPEG compression).

Solution a. provide the best picture quality. Since the input data is 24-bit RGB equivalent. It could converted to RGB888 for LCD display. Solution b. provide the worst picture quality. Since the input data is only 16-bit RGB565, even it is converted to YCbCr, the color depth is still 16-bit. The solution c. provide similar picture quality as solution a. But since preview is RGB565, capture is YCbCr, preview picture may looks a little different than captured picture.

1.4 Equations to Convert from One Format to Another

YCbCr to RGB24

$Y = 0.299R + 0.587G + 0.114B$

$Cb = 0.568(B-Y) + 128 = –0.172R – 0.339G + 0.511B + 128$

$Cr = 0.713(R-Y) + 128 = 0.511R – 0.428G – 0.083B + 128$

```
Y = ((77 * R + 150 * G + 29 * B) >> 8);
Cb = ((-43 * R - 85 * G + 128 * B) >> 8) + 128;
Cr = ((128 * R - 107 * G - 21 * B) >> 8) + 128;
```

RGB24 to YcbCr

$R = Y + 1.371(Cr – 128)$

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

B = Y + 1.732(Cb – 128)

R = Y + (351*(Cr – 128)) >> 8

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

B = Y + (443*(Cb – 128)) >> 8

2. How to Select Output Resolution?

2.1 back-end with ISP

If back-end chip has built-in ISP (Full ISP or YCbCr ISP), the ISP could do image scale. So OV2640 outputs only SVGA format for preview and UXGA for capture. ISP scaled SVGA image to other resolution that mobile device needed for LCD display. And the ISP scaled UXGA image to other resolution that the mobile device needed for capture.

2.2 back-end without ISP

If back-end chip doesn't have image scale capability, then the LCD scaler of OV2640 must be used to scale output resolution exactly the LCD size. For example, if the LCD size is 176x220, then the LCD scaler will scale the output size to 176x220.

In this case, OV2640 output small resolution for preview, and several other resolution for capture. The resolution for capture may include: QQVGA, QVGA, QCIF, CIF, VGA, SVGA,SXGA,UXGA.

3. How to Adjust frame rate

The recommended frame rates are 30fps and 15fps preview for 60Hz light environment, 25fps and 14.3fps preview for 50Hz light environment. The recommended frame rate for capture is 7.5fps for 60hz light environment and 7.14fps for 50hz light environment. The frame rate for night mode is lower, we'll discuss night mode later.

Reference settings for above frame rates are listed below.

3.1 SVGA Preview, 30fps, 24Mhz input clock

```
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x00);
write_SCCB(0x12, 0x40);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x46, 0x00);
write_SCCB(0x47, 0x00);
write_SCCB(0x3d, 0x38);
```

3.2 SVGA Preview, 15fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x01);
write_SCCB(0x12, 0x40);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x46, 0x00);
write_SCCB(0x47, 0x00);
write_SCCB(0x3d, 0x38);

3.3 SVGA Preview, 25fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x00);
write_SCCB(0x12, 0x40);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x46, 0x87);
write_SCCB(0x47, 0x00);
write_SCCB(0x3d, 0x38);

3.4 SVGA Preview, 14.3fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x01);
write_SCCB(0x12, 0x40);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x46, 0x22);
write_SCCB(0x47, 0x00);
write_SCCB(0x3d, 0x38);

3.5 UXGA Capture, 7.5fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x01);
write_SCCB(0x12, 0x00);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x46, 0x00);
write_SCCB(0x47, 0x00);
write_SCCB(0x3d, 0x34);

3.6 UXGA Capture, 7.14fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x01);
write_SCCB(0x12, 0x00);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x46, 0x3f);
write_SCCB(0x47, 0x00);
write_SCCB(0x3d, 0x34);

4. How to set Night Mode Preview

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 3.75fps.

4.1 Night Mode with Fixed Frame Rate

3.75fps night mode for 60Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x07);

3.125fps night mode for 50Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x07);

4.2 Night Mode with Auto Frame Rate

30fps ~ 3.75fps night mode for 60Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x00);
write_SCCB(0x0f, 0x4b);
write_SCCB(0x03, 0xcf);

15fps ~ 3.75fps night mode for 60Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x01);
write_SCCB(0x0f, 0x4b);
write_SCCB(0x03, 0x8f);

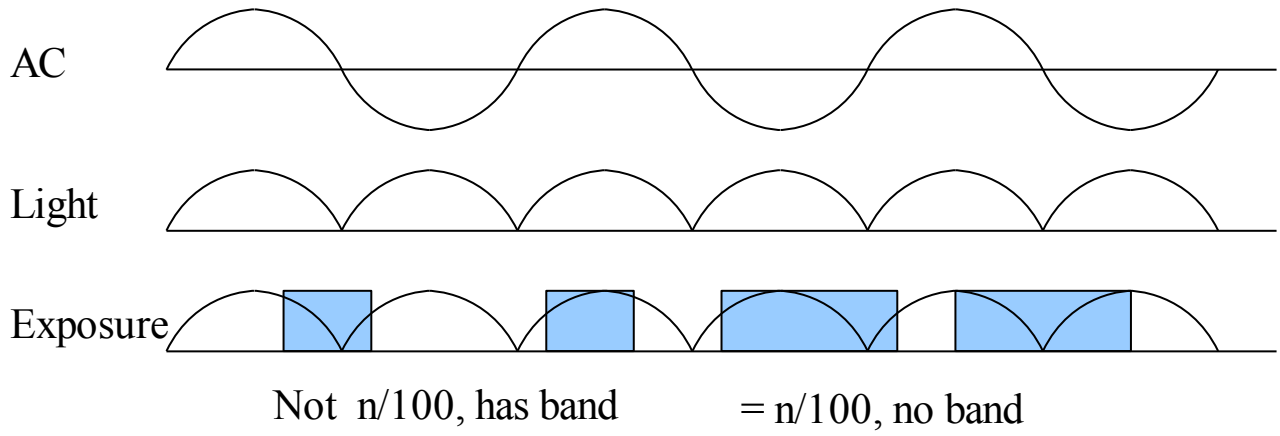
25fps ~ 3.125fps night mode for 50Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;

write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x00);
write_SCCB(0x0f, 0x4b);
write_SCCB(0x03, 0xcf);

14.3fps ~ 3.6fps night mode for 50Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x11, 0x01);
write_SCCB(0x0f, 0x4b);
write_SCCB(0x03, 0x8f);

5. How to Remove Light Band in Preview Mode

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 OV2640 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 30fps SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x38); //select 60hz banding filter ,bit[1]control auto, set 0, close auto banding
write_SCCB(0x4f, 0xca); //50Hz banding filter
write_SCCB(0x50, 0xa8); //60Hz banding filter
write_SCCB(0x5a, 0x23); //3 step for 50hz,4 step for 60hz

banding filter setting for 15fps SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x38); //select 60hz banding filter ,bit[1]control auto, set 0, close auto banding
write_SCCB(0x4f, 0x65); //50Hz banding filter
write_SCCB(0x50, 0x54); //60Hz banding filter
write_SCCB(0x5a, 0x57); //6 step for 50hz,8step for 60hz

banding filter setting for 25fps SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x3c); //select 50hz banding filter ,bit[1]control auto, set 0, close auto banding
write_SCCB(0x4f, 0xa8); //50Hz banding filter
write_SCCB(0x50, 0x8c); //60Hz banding filter
write_SCCB(0x5a, 0x33); //4 step for 50hz,4 step for 60hz

banding filter setting for 14.3fps SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x3c); //select 50hz banding filter ,bit[1]control auto, set 0, close auto banding
write_SCCB(0x4f, 0x54); //50Hz banding filter

write_SCCB(0x50, 0x46); //60Hz banding filter
write_SCCB(0x5a, 0x78); //8 step for 50hz,9step for 60hz

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. OV2640 could automatic select 50Hz or 60Hz banding filter based on light frequency detection.

Automatic select banding filter for 30fps SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x3a); //Automatically select banding filter
write_SCCB(0x4f, 0xca); //50Hz banding filter
write_SCCB(0x50, 0xa8); //60Hz banding filter
write_SCCB(0x5a, 0x23); //3 step for 50hz,4 step for 60hz

Automatic select banding filter for 15fsp SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x3a); //Automatically select banding filter
write_SCCB(0x4f, 0x65); //50Hz banding filter
write_SCCB(0x50, 0x54); //60Hz banding filter
write_SCCB(0x5a, 0x57); //6 step for 50hz,8step for 60hz

Automatic select banding filter for 25fps SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x3a); //Automatically select banding filter
write_SCCB(0x4f, 0xa8); //50Hz banding filter
write_SCCB(0x50, 0x8c); //60Hz banding filter
write_SCCB(0x5a, 0x33); //4 step for 50hz,4 step for 60hz

Automatic select banding filter for 14.3fsp SVGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x01); //select bank1
write_SCCB(0x13, 0xe5); bit[5]enable banding filter
write_SCCB(0x0c, 0x3a); //Automatically select banding filter
write_SCCB(0x4f, 0x60); //50Hz banding filter
write_SCCB(0x50, 0x50); //60Hz banding filter
write_SCCB(0x5a, 0x67); //7 step for 50hz,8step for 60hz

5.5 Remove Light Band in Capture

If capture uses the same resolution and frame rate as preview, the light band is removed in

capture. If capture uses different resolution or different frame rate as preview, then the light band should be removed in capture by exposure calculation. Please read section 14.7.3.

5.6 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 OV2640. So there is no way to remove light band in this condition.

6. White Balance

OV2640 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.

Settings

```
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00); //Select bank0
write_SCCB(0xc7, 0x10); // 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

Disadvantage of Advanced AWB

Advanced white balance setting is depend on lens. The setting must be adjusted for every module with new lens. The adjustment must be done by OmniVision FAE in optical lab with some optical equipment such as light box, color checker etc.

Settings

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.

OV2640 has built-in defect pixel correction function. If OV2640 output YCbCr, RGB565, 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 back-end chip should be used instead.

Please pay attention to the defect pixel correction function of back-end chip. Some back-end chip may not be able to correct all defect pixels of OV2640.

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 OV2640. 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 OV2640 output image smaller than SVGA, it may support continuous digital zoom. For example

UXGA	no digital zoom supported
SVGA	1-2x
VGA	1-2.5x
QVGA	1-5x

If back-end chip support scale up, then more zoom level could be supported.

11. OV2640 Functions

11.1 Light Mode

Auto
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00)
write_SCCB(0xc7, 0x00); //AWB on

Sunny
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00)
write_SCCB(0xc7, 0x40); //AWB off
write_SCCB(0xcc, 0x5e);
write_SCCB(0xcd, 0x41);
write_SCCB(0xce, 0x54);

Cloudy
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00)
write_SCCB(0xc7, 0x40); //AWB off
write_SCCB(0xcc, 0x65);
write_SCCB(0xcd, 0x41);
write_SCCB(0xce, 0x4f);

Office
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00)
write_SCCB(0xc7, 0x40); //AWB off
write_SCCB(0xcc, 0x52);
write_SCCB(0xcd, 0x41);
write_SCCB(0xce, 0x66);

Home
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00)

```
write_SCCB(0xc7, 0x40); //AWB off
write_SCCB(0xcc, 0x42);
write_SCCB(0xcd, 0x3f);
write_SCCB(0xce, 0x71);
```

11.2 Color Saturation

The color saturation of OV2640 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 + 2
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x02);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x68);
write_SCCB(0x7d, 0x68);


Saturation + 1
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x02);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x58);
write_SCCB(0x7d, 0x58);

Saturation 0
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x02);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x48);
write_SCCB(0x7d, 0x48);

Saturation -1
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x02);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x38);
write_SCCB(0x7d, 0x38);
```

Saturation - 2
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x02);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x28);
write_SCCB(0x7d, 0x28);

11.3 Brightness

The brightness of OV2640 could be adjusted. Higher brightness will make the picture more bright. The side effect of higher brightness is the picture looks foggy.

Brightness +2
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x09);
write_SCCB(0x7d, 0x40);
write_SCCB(0x7d, 0x00);

Brightness +1
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x09);
write_SCCB(0x7d, 0x30);
write_SCCB(0x7d, 0x00);

Brightness 0
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x09);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x00);

Brightness -1
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x09);

write_SCCB(0x7d, 0x10);
write_SCCB(0x7d, 0x00);

Brightness -2
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x09);
write_SCCB(0x7d, 0x00);
write_SCCB(0x7d, 0x00);

11.4 Contrast

The contrast of OV2640 could be adjusted. Higher contrast will make the picture sharp. But the side effect is loosing dynamic range.

Contrast +2
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x07);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x28);
write_SCCB(0x7d, 0x0c);
write_SCCB(0x7d, 0x06);

Contrast +1
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x07);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x24);
write_SCCB(0x7d, 0x16);
write_SCCB(0x7d, 0x06);

Contrast 0
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x07);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x20);

write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x06);

Contrast -1
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x07);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x1c);
write_SCCB(0x7d, 0x2a);
write_SCCB(0x7d, 0x06);

Contrast -2
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x04);
write_SCCB(0x7c, 0x07);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x18);
write_SCCB(0x7d, 0x34);
write_SCCB(0x7d, 0x06);

11.5 Special effects

OV2640 support some special effects such as B/W, negative, sepia, bluish, reddish, greenish etc. If users need other special effects, it should be supported by back-end chips.

Antique
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x18);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0x40);
write_SCCB(0x7d, 0xa6);

Bluish
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x18);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0xa0);
write_SCCB(0x7d, 0x40);

Greenish
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x18);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0x40);
write_SCCB(0x7d, 0x40);

Reddish
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x18);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0x40);
write_SCCB(0x7d, 0xc0);

B&W
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x18);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0x80);
write_SCCB(0x7d, 0x80);

Negative
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x40);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0x80);
write_SCCB(0x7d, 0x80;

B&W negative
SCCB_salve_Address = 0x60;
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x58);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0x80);
write_SCCB(0x7d, 0x80;

Normal
SCCB_salve_Address = 0x60;

```
write_SCCB(0xff, 0x00);
write_SCCB(0x7c 0x00);
write_SCCB(0x7d, 0x00);
write_SCCB(0x7c 0x05);
write_SCCB(0x7d, 0x80);
write_SCCB(0x7d, 0x80;
```

11.6 YUV Sequence

Y U Y V

```
SCCB_slave_address = 0x60;
write_SCCB(0xff, 0x00);

temp = read_SCCB(0xda);    //register  0xda bit 0 to '0'
temp &= 0xfe;
write_SCCB(0xda, temp);

temp = read_SCCB(0xda);    //register  0xc2 bit 4 to '0'
temp &= 0xef;
write_SCCB(0xda, temp);
```

Y V Y U

```
SCCB_slave_address = 0x60;
write_SCCB(0xff, 0x00);

temp = read_SCCB(0xda);    //register  0xda bit 0 to '0'
temp &= 0xfe;
write_SCCB(0xda, temp);

temp = read_SCCB(0xda);    //register  0xc2 bit 4 to '1'
temp |= 0x10;
write_SCCB(0xda, temp);
```

V Y U Y

```
SCCB_slave_address = 0x60;
write_SCCB(0xff, 0x00);

temp = read_SCCB(0xda);    //register  0xda bit 0 to '1'
temp |= 0x01;
write_SCCB(0xda, temp);

temp = read_SCCB(0xda);    //register  0xc2 bit 4 to '0'
temp &= 0xef;
write_SCCB(0xda, temp);
```

U Y V Y
SCCB_slave_address = 0x60;
write_SCCB(0xff, 0x00);

temp = read_SCCB(0xda); //register 0xda bit 0 to '1'
temp L= 0x01;
write_SCCB(0xda, temp);

temp = read_SCCB(0xda); //register 0xc2 bit 4 to '0'
temp |= 0x10;
write_SCCB(0xda, 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 OV2640 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 OV2640 the focus distance is about 120~150cm. The depth of field is about from 60~75cm to infinite. If checking resolution of camera module, the resolution chart should be placed 120~150 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 fogy. 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.

13. Reference Settings

13.1 YCbCr Reference Setting

13.1.1 SVGA Preview

```
// OV2640_SVGA_YUV_AM 14.3 fps
// 24 MHz input clock
//
//
SCCB_slave_address = 0x60;
write_SCCB(0xff, 0x01);
write_SCCB(0x12, 0x80);
delay(1ms);
write_SCCB(0xff, 0x00);
write_SCCB(0x2c, 0xff);
write_SCCB(0x2e, 0xdf);
write_SCCB(0xff, 0x01);
write_SCCB(0x3c, 0x32);
//
write_SCCB(0x11, 0x01);
write_SCCB(0x09, 0x02);
write_SCCB(0x04, 0x28);
write_SCCB(0x13, 0xe5);
write_SCCB(0x14, 0x48);
write_SCCB(0x2c, 0x0c);
write_SCCB(0x33, 0x78);
write_SCCB(0x3a, 0x33);
write_SCCB(0x3b, 0xfB);
//
write_SCCB(0x3e, 0x00);
write_SCCB(0x43, 0x11);
write_SCCB(0x16, 0x10);
//
write_SCCB(0x39, 0x92);
//
write_SCCB(0x35, 0xda);
write_SCCB(0x22, 0x1a);
write_SCCB(0x37, 0xc3);
write_SCCB(0x23, 0x00);
write_SCCB(0x34, 0xc0);
write_SCCB(0x36, 0x1a);
write_SCCB(0x06, 0x88);
```

```
write_SCCB(0x07, 0xc0);
write_SCCB(0x0d, 0x87);
write_SCCB(0x0e, 0x41);
write_SCCB(0x4c, 0x00);
write_SCCB(0x48, 0x00);
write_SCCB(0x5B, 0x00);
write_SCCB(0x42, 0x03);
//
write_SCCB(0x4a, 0x81);
write_SCCB(0x21, 0x99);
//
write_SCCB(0x24, 0x40);
write_SCCB(0x25, 0x38);
write_SCCB(0x26, 0x82);
write_SCCB(0x5c, 0x00);
write_SCCB(0x63, 0x00);
write_SCCB(0x61, 0x70);
write_SCCB(0x62, 0x80);
write_SCCB(0x7c, 0x05);
//
write_SCCB(0x20, 0x80);
write_SCCB(0x28, 0x30);
write_SCCB(0x6c, 0x00);
write_SCCB(0x6d, 0x80);
write_SCCB(0x6e, 0x00);
write_SCCB(0x70, 0x02);
write_SCCB(0x71, 0x94);
write_SCCB(0x73, 0xc1);
//
write_SCCB(0x12, 0x40);
write_SCCB(0x17, 0x11);
write_SCCB(0x18, 0x43);
write_SCCB(0x19, 0x00);
write_SCCB(0x1a, 0x4b);
write_SCCB(0x32, 0x09);
write_SCCB(0x37, 0xc0);
write_SCCB(0x4f, 0x60);
write_SCCB(0x50, 0xa8);
write_SCCB(0x6d, 0x00);
write_SCCB(0x3d, 0x38);
//
write_SCCB(0x46, 0x3f);
write_SCCB(0x4f, 0x60);
write_SCCB(0x0c, 0x3c);
//
write_SCCB(0xff, 0x00);
write_SCCB(0xe5, 0x7f);
write_SCCB(0xf9, 0xc0);
```

```
write_SCCB(0x41, 0x24);
write_SCCB(0xe0, 0x14);
write_SCCB(0x76, 0xff);
write_SCCB(0x33, 0xa0);
write_SCCB(0x42, 0x20);
write_SCCB(0x43, 0x18);
write_SCCB(0x4c, 0x00);
write_SCCB(0x87, 0xd5);
write_SCCB(0x88, 0x3f);
write_SCCB(0xd7, 0x03);
write_SCCB(0xd9, 0x10);
write_SCCB(0xd3, 0x82);
//
write_SCCB(0xc8, 0x08);
write_SCCB(0xc9, 0x80);
//
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x00);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x48);
write_SCCB(0x7d, 0x48);
write_SCCB(0x7c, 0x08);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x10);
write_SCCB(0x7d, 0x0e);
//
write_SCCB(0x90, 0x00);
write_SCCB(0x91, 0x0e);
write_SCCB(0x91, 0x1a);
write_SCCB(0x91, 0x31);
write_SCCB(0x91, 0x5a);
write_SCCB(0x91, 0x69);
write_SCCB(0x91, 0x75);
write_SCCB(0x91, 0x7e);
write_SCCB(0x91, 0x88);
write_SCCB(0x91, 0x8f);
write_SCCB(0x91, 0x96);
write_SCCB(0x91, 0xa3);
write_SCCB(0x91, 0xaf);
write_SCCB(0x91, 0xc4);
write_SCCB(0x91, 0xd7);
write_SCCB(0x91, 0xe8);
write_SCCB(0x91, 0x20);
//
write_SCCB(0x92, 0x00);
write_SCCB(0x93, 0x06);
write_SCCB(0x93, 0xe3);
write_SCCB(0x93, 0x05);
```

```
write_SCCB(0x93, 0x05);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x04);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
//
write_SCCB(0x96, 0x00);
write_SCCB(0x97, 0x08);
write_SCCB(0x97, 0x19);
write_SCCB(0x97, 0x02);
write_SCCB(0x97, 0x0c);
write_SCCB(0x97, 0x24);
write_SCCB(0x97, 0x30);
write_SCCB(0x97, 0x28);
write_SCCB(0x97, 0x26);
write_SCCB(0x97, 0x02);
write_SCCB(0x97, 0x98);
write_SCCB(0x97, 0x80);
write_SCCB(0x97, 0x00);
write_SCCB(0x97, 0x00);
//
write_SCCB(0xc3, 0xed);
write_SCCB(0xa4, 0x00);
write_SCCB(0xa8, 0x00);
write_SCCB(0xc5, 0x11);
write_SCCB(0xc6, 0x51);
write_SCCB(0xbf, 0x80);
write_SCCB(0xc7, 0x10);
write_SCCB(0xb6, 0x66);
write_SCCB(0xb8, 0xA5);
write_SCCB(0xb7, 0x64);
write_SCCB(0xb9, 0x7C);
write_SCCB(0xb3, 0xaf);
write_SCCB(0xb4, 0x97);
write_SCCB(0xb5, 0xFF);
write_SCCB(0xb0, 0xC5);
write_SCCB(0xb1, 0x94);
write_SCCB(0xb2, 0x0f);
write_SCCB(0xc4, 0x5c);
//
write_SCCB(0xc0, 0x64);
write_SCCB(0xc1, 0x4B);
write_SCCB(0x8c, 0x00);
```

```
write_SCCB(0x86, 0x3D);
write_SCCB(0x50, 0x00);
write_SCCB(0x51, 0xC8);
write_SCCB(0x52, 0x96);
write_SCCB(0x53, 0x00);
write_SCCB(0x54, 0x00);
write_SCCB(0x55, 0x00);
write_SCCB(0x5a, 0xC8);
write_SCCB(0x5b, 0x96);
write_SCCB(0x5c, 0x00);
write_SCCB(0xd3, 0x82);
//
write_SCCB(0xc3, 0xed);
write_SCCB(0x7f, 0x00);
//
write_SCCB(0xda, 0x00);
//
write_SCCB(0xe5, 0x1f);
write_SCCB(0xe1, 0x67);
write_SCCB(0xe0, 0x00);
write_SCCB(0xdd, 0x7f);
write_SCCB(0x05, 0x00);
```

13.1.2 UXGA Capture

```
// OV2640_UXGA_YUV_AM 7.5 fps
// 24 MHz input clock
//
SCCB_slave_address = 0x60;
write_SCCB(0xff, 0x01);

write_SCCB(0x12, 0x80);
delay(1ms)
write_SCCB(0xff, 0x00);
write_SCCB(0x2c, 0xff);
write_SCCB(0x2e, 0xdf);
write_SCCB(0xff, 0x01);
write_SCCB(0x3c, 0x32);
//
write_SCCB(0x11, 0x01);
write_SCCB(0x09, 0x02);
write_SCCB(0x04, 0x28);
write_SCCB(0x13, 0xe5);
write_SCCB(0x14, 0x48);
write_SCCB(0x2c, 0x0c);
write_SCCB(0x33, 0x78);
write_SCCB(0x3a, 0x33);
write_SCCB(0x3b, 0xfB);
//
```

```
write_SCCB(0x3e, 0x00);
write_SCCB(0x43, 0x11);
write_SCCB(0x16, 0x10);
//
write_SCCB(0x39, 0x82);
//
write_SCCB(0x35, 0x88);
write_SCCB(0x22, 0x0a);
write_SCCB(0x37, 0x40);
write_SCCB(0x23, 0x00);
write_SCCB(0x34, 0xa0);
write_SCCB(0x36, 0x1a);
write_SCCB(0x06, 0x02);
write_SCCB(0x07, 0xc0);
write_SCCB(0x0d, 0xb7);
write_SCCB(0x0e, 0x01);
write_SCCB(0x4c, 0x00);
write_SCCB(0x48, 0x00);
write_SCCB(0x5B, 0x00);
write_SCCB(0x42, 0x83);
//
write_SCCB(0x4a, 0x81);
write_SCCB(0x21, 0x99);
//
write_SCCB(0x24, 0x40);
write_SCCB(0x25, 0x38);
write_SCCB(0x26, 0x82);
write_SCCB(0x5c, 0x00);
write_SCCB(0x63, 0x00);
write_SCCB(0x46, 0x00);
write_SCCB(0x0c, 0x38);
//
write_SCCB(0x61, 0x70);
write_SCCB(0x62, 0x80);
write_SCCB(0x7c, 0x05);
//
write_SCCB(0x20, 0x80);
write_SCCB(0x28, 0x30);
write_SCCB(0x6c, 0x00);
write_SCCB(0x6d, 0x80);
write_SCCB(0x6e, 0x00);
write_SCCB(0x70, 0x02);
write_SCCB(0x71, 0x94);
write_SCCB(0x73, 0xc1);
//
write_SCCB(0x3d, 0x34);
write_SCCB(0x5a, 0x57);
write_SCCB(0x4f, 0xbb);
```

```
write_SCCB(0x50, 0x9c);
//
//
write_SCCB(0xff, 0x00);
write_SCCB(0xe5, 0x7f);
write_SCCB(0xf9, 0xc0);
write_SCCB(0x41, 0x24);
write_SCCB(0xe0, 0x14);
write_SCCB(0x76, 0xff);
write_SCCB(0x33, 0xa0);
write_SCCB(0x42, 0x20);
write_SCCB(0x43, 0x18);
write_SCCB(0x4c, 0x00);
write_SCCB(0x87, 0xd0);
write_SCCB(0x88, 0x3f);
write_SCCB(0xd7, 0x03);
write_SCCB(0xd9, 0x10);
write_SCCB(0xd3, 0x82);
//
write_SCCB(0xc8, 0x08);
write_SCCB(0xc9, 0x80);
//
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x00);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x48);
write_SCCB(0x7d, 0x48);
write_SCCB(0x7c, 0x08);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x10);
write_SCCB(0x7d, 0x0e);
//
write_SCCB(0x90, 0x00);
write_SCCB(0x91, 0x0e);
write_SCCB(0x91, 0x1a);
write_SCCB(0x91, 0x31);
write_SCCB(0x91, 0x5a);
write_SCCB(0x91, 0x69);
write_SCCB(0x91, 0x75);
write_SCCB(0x91, 0x7e);
write_SCCB(0x91, 0x88);
write_SCCB(0x91, 0x8f);
write_SCCB(0x91, 0x96);
write_SCCB(0x91, 0xa3);
write_SCCB(0x91, 0xaf);
write_SCCB(0x91, 0xc4);
write_SCCB(0x91, 0xd7);
write_SCCB(0x91, 0xe8);
```

```
write_SCCB(0x91, 0x20);
//
write_SCCB(0x92, 0x00);
write_SCCB(0x93, 0x06);
write_SCCB(0x93, 0xe3);
write_SCCB(0x93, 0x05);
write_SCCB(0x93, 0x05);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x04);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
//
write_SCCB(0x96, 0x00);
write_SCCB(0x97, 0x08);
write_SCCB(0x97, 0x19);
write_SCCB(0x97, 0x02);
write_SCCB(0x97, 0x0c);
write_SCCB(0x97, 0x24);
write_SCCB(0x97, 0x30);
write_SCCB(0x97, 0x28);
write_SCCB(0x97, 0x26);
write_SCCB(0x97, 0x02);
write_SCCB(0x97, 0x98);
write_SCCB(0x97, 0x80);
write_SCCB(0x97, 0x00);
write_SCCB(0x97, 0x00);
//
write_SCCB(0xc3, 0xed);
write_SCCB(0xc4, 0x9a);
write_SCCB(0xa4, 0x00);
write_SCCB(0xa8, 0x00);
write_SCCB(0xc5, 0x11);
write_SCCB(0xc6, 0x51);
write_SCCB(0xbf, 0x80);
write_SCCB(0xc7, 0x10);
write_SCCB(0xb6, 0x66);
write_SCCB(0xb8, 0xA5);
write_SCCB(0xb7, 0x64);
write_SCCB(0xb9, 0x7C);
write_SCCB(0xb3, 0xaf);
write_SCCB(0xb4, 0x97);
write_SCCB(0xb5, 0xFF);
write_SCCB(0xb0, 0xC5);
```

```
write_SCCB(0xb1, 0x94);
write_SCCB(0xb2, 0x0f);
write_SCCB(0xc4, 0x5c);
//
write_SCCB(0xc0, 0xc8);
write_SCCB(0xc1, 0x96);
write_SCCB(0x86, 0x1d);
write_SCCB(0x50, 0x00);
write_SCCB(0x51, 0x90);
write_SCCB(0x52, 0x2c);
write_SCCB(0x53, 0x00);
write_SCCB(0x54, 0x00);
write_SCCB(0x55, 0x88);
write_SCCB(0x57, 0x00);
write_SCCB(0x5a, 0x90);
write_SCCB(0x5b, 0x2c);
write_SCCB(0x5c, 0x05);
//
write_SCCB(0xc3, 0xed);
write_SCCB(0x7f, 0x00);
//
write_SCCB(0xda, 0x00);
//
write_SCCB(0xe5, 0x1f);
write_SCCB(0xe1, 0x67);
write_SCCB(0xe0, 0x00);
write_SCCB(0xdd, 0x7f);
write_SCCB(0x05, 0x00);
```

13.2 RGB 565 Reference Setting

```
//MCLK 24Mhz, SVGA RGB565 output 25fps
SCCB_slave_address = 0x60;

write_SCCB(0xff, 0x01);
write_SCCB(0x12, 0x80);
delay(5ms);
write_SCCB(0xff, 0x00);
write_SCCB(0x2c, 0xff);
write_SCCB(0x2e, 0xdf);
write_SCCB(0xff, 0x01);
write_SCCB(0x3c, 0x32);
//
write_SCCB(0x11, 0x00);
write_SCCB(0x09, 0x02);
write_SCCB(0x04, 0x28);
write_SCCB(0x13, 0xe5);
write_SCCB(0x14, 0x48);
```

```
write_SCCB(0x2c, 0x0c);
write_SCCB(0x33, 0x78);
write_SCCB(0x3a, 0x33);
write_SCCB(0x3b, 0xfB);
//
write_SCCB(0x3e, 0x00);
write_SCCB(0x43, 0x11);
write_SCCB(0x16, 0x10);
//
write_SCCB(0x39, 0x92);
//
write_SCCB(0x35, 0xda);
write_SCCB(0x22, 0x1a);
write_SCCB(0x37, 0xc3);
write_SCCB(0x23, 0x00);
write_SCCB(0x34, 0xc0);
write_SCCB(0x36, 0x1a);
write_SCCB(0x06, 0x88);
write_SCCB(0x07, 0xc0);
write_SCCB(0x0d, 0x87);
write_SCCB(0x0e, 0x41);
write_SCCB(0x4c, 0x00);
write_SCCB(0x48, 0x00);
write_SCCB(0x5B, 0x00);
write_SCCB(0x42, 0x03);
//
write_SCCB(0x4a, 0x81);
write_SCCB(0x21, 0x99);
//
write_SCCB(0x24, 0x40);
write_SCCB(0x25, 0x38);
write_SCCB(0x26, 0x82);
write_SCCB(0x5c, 0x00);
write_SCCB(0x63, 0x00);
write_SCCB(0x46, 0x22);
write_SCCB(0x0c, 0x3c);
//
write_SCCB(0x61, 0x70);
write_SCCB(0x62, 0x80);
write_SCCB(0x7c, 0x05);
//
write_SCCB(0x20, 0x80);
write_SCCB(0x28, 0x30);
write_SCCB(0x6c, 0x00);
write_SCCB(0x6d, 0x80);
write_SCCB(0x6e, 0x00);
write_SCCB(0x70, 0x02);
write_SCCB(0x71, 0x94);
```

```
write_SCCB(0x73, 0xc1);
//
write_SCCB(0x12, 0x40);
write_SCCB(0x17, 0x11);
write_SCCB(0x18, 0x43);
write_SCCB(0x19, 0x00);
write_SCCB(0x1a, 0x4b);
write_SCCB(0x32, 0x09);
write_SCCB(0x37, 0xc0);
write_SCCB(0x4f, 0xca);
write_SCCB(0x50, 0xa8);
write_SCCB(0x5a, 0x23);
write_SCCB(0x6d, 0x00);
write_SCCB(0x3d, 0x38);
//
write_SCCB(0xff, 0x00);
write_SCCB(0xe5, 0x7f);
write_SCCB(0xf9, 0xc0);
write_SCCB(0x41, 0x24);
write_SCCB(0xe0, 0x14);
write_SCCB(0x76, 0xff);
write_SCCB(0x33, 0xa0);
write_SCCB(0x42, 0x20);
write_SCCB(0x43, 0x18);
write_SCCB(0x4c, 0x00);
write_SCCB(0x87, 0xd5);
write_SCCB(0x88, 0x3f);
write_SCCB(0xd7, 0x03);
write_SCCB(0xd9, 0x10);
write_SCCB(0xd3, 0x82);
//
write_SCCB(0xc8, 0x08);
write_SCCB(0xc9, 0x80);
//
write_SCCB(0x7c, 0x00);
write_SCCB(0x7d, 0x00);
write_SCCB(0x7c, 0x03);
write_SCCB(0x7d, 0x48);
write_SCCB(0x7d, 0x48);
write_SCCB(0x7c, 0x08);
write_SCCB(0x7d, 0x20);
write_SCCB(0x7d, 0x10);
write_SCCB(0x7d, 0x0e);
//
write_SCCB(0x90, 0x00);
write_SCCB(0x91, 0x0e);
write_SCCB(0x91, 0x1a);
write_SCCB(0x91, 0x31);
```

```
write_SCCB(0x91, 0x5a);
write_SCCB(0x91, 0x69);
write_SCCB(0x91, 0x75);
write_SCCB(0x91, 0x7e);
write_SCCB(0x91, 0x88);
write_SCCB(0x91, 0x8f);
write_SCCB(0x91, 0x96);
write_SCCB(0x91, 0xa3);
write_SCCB(0x91, 0xaf);
write_SCCB(0x91, 0xc4);
write_SCCB(0x91, 0xd7);
write_SCCB(0x91, 0xe8);
write_SCCB(0x91, 0x20);
//
write_SCCB(0x92, 0x00);
write_SCCB(0x93, 0x06);
write_SCCB(0x93, 0xe3);
write_SCCB(0x93, 0x05);
write_SCCB(0x93, 0x05);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x04);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
write_SCCB(0x93, 0x00);
//
write_SCCB(0x96, 0x00);
write_SCCB(0x97, 0x08);
write_SCCB(0x97, 0x19);
write_SCCB(0x97, 0x02);
write_SCCB(0x97, 0x0c);
write_SCCB(0x97, 0x24);
write_SCCB(0x97, 0x30);
write_SCCB(0x97, 0x28);
write_SCCB(0x97, 0x26);
write_SCCB(0x97, 0x02);
write_SCCB(0x97, 0x98);
write_SCCB(0x97, 0x80);
write_SCCB(0x97, 0x00);
write_SCCB(0x97, 0x00);
//
write_SCCB(0xc3, 0xed);
write_SCCB(0xa4, 0x00);
write_SCCB(0xa8, 0x00);
write_SCCB(0xc5, 0x11);
```

```
write_SCCB(0xc6, 0x51);
write_SCCB(0xbf, 0x80);
write_SCCB(0xc7, 0x10);
write_SCCB(0xb6, 0x66);
write_SCCB(0xb8, 0xA5);
write_SCCB(0xb7, 0x64);
write_SCCB(0xb9, 0x7C);
write_SCCB(0xb3, 0xaf);
write_SCCB(0xb4, 0x97);
write_SCCB(0xb5, 0xFF);
write_SCCB(0xb0, 0xC5);
write_SCCB(0xb1, 0x94);
write_SCCB(0xb2, 0x0f);
write_SCCB(0xc4, 0x5c);
//
write_SCCB(0xc0, 0x64);
write_SCCB(0xc1, 0x4B);
write_SCCB(0x8c, 0x00);
write_SCCB(0x86, 0x3D);
write_SCCB(0x50, 0x00);
write_SCCB(0x51, 0xC8);
write_SCCB(0x52, 0x96);
write_SCCB(0x53, 0x00);
write_SCCB(0x54, 0x00);
write_SCCB(0x55, 0x00);
write_SCCB(0x5a, 0xC8);
write_SCCB(0x5b, 0x96);
write_SCCB(0x5c, 0x00);
write_SCCB(0xd3, 0x82);
//
write_SCCB(0xc3, 0xed);
write_SCCB(0x7f, 0x00);
//
write_SCCB(0xda, 0x08);
//
write_SCCB(0xe5, 0x1f);
write_SCCB(0xe1, 0x67);
write_SCCB(0xe0, 0x00);
write_SCCB(0xdd, 0x7f);
write_SCCB(0x05, 0x00);
```

13.3 RGB raw Reference Setting

14. Capture Sequence

14.1 Shutter

The shutter of OV2640 controls exposure time. The unit of shutter is line period.

Shutter value has limitation for each output resolution. If no dummy lines are inserted, the maximum shutter value for QXGA resolution is 1248. The maximum shutter value for XGA resolution is 672.

Default_XGA_maximum_shutter = 672;
Default_QXGA_maximum_shutter = 1248;

The shutter value are stored in 3 registers, reg0x45, reg0x10 and reg0x04 of bank 1.

Shutter = (reg0x45 & 0x3f) << 10 + reg0x10<<2 + (reg0x04 & 0x03);

14. 2 Dummy Lines

The exposure could be increased further by insert dummy lines. When dummy lines are inserted, frame rate also changes.

The are 2 kinds of dummy lines could be inserted. Dummy line before data output and dummy line after data output.

14.2.1 Extra Line

If dummy lines are inserted before data output, which is called extra line, the actual exposure time is increased. The extra line is controlled by register 0x2d and 0x2e of bank 1.

Exposure = Shutter + Extra_lines

Extra_lines = reg0x2d + (reg0x2e << 8);

the maximum shutter value is not changed.

So even shutter value is 0, the minimum exposure time is Extra_lines.

Usually, extra lines should be inserted automatically. If the environment is dark, longer exposure time is required, extra line number increased. If the environment is bright, shorter exposure time is required, extra line number decreased. If extra line is inserted manually and the value is fixed, the total exposure time can not be less then extra_line in bright environment, the image would be over exposed.

The extra lines are inserted inside the active period of Vsync, the timing of output period in which Vsync is inactive is not changed.

14.2.2 Dummy Line

If dummy lines are inserted after data output, which is called dummy line, the maximum shutter

value is changed. The dummy lines are inserted between two Vsync. The number of dummy lines is controlled by register 0x46 and 0x47 of bank 1.
SVGA_maximum_shutter = Default_SVGA_Maximum_Shutter + Dummy_line
UXGA_maximum_shutter = Default_UXGA_Maximum_Shutter + Dummy_line

The exposure time is

Exposure = Shutter

Dummy_line = Reg0x47 <<8 + Reg0x46;

	Extra line	Dummy Line
Registers	0x2d, 0x2e	0x46, 0x47
Minimum Shutter Value	0	0
Maximum Shutter Value	1248 for UXGA 672 for SVGA	1248 + Dummy_line for UXGA 672 + Dummy_line for SVGA
Minimum Exposure	Extra_lines	0
Maximum Exposure	Maximum_Shutter + Extra_lines	Maximum_Shutter

So if both dummy line and extra line are inserted, the exposure time is

Exposure = Shutter + Extra_Lines

And the maximum shutter value is

SVGA_maximum_shutter = Default_SVGA_Maximum_Shutter + Dummy_line
UXGA_maximum_shutter = Default_UXGA_Maximum_Shutter + Dummy_line

14. 3 Dummy Pixels

If no dummy pixel is inserted, the line width is called default line width.
Default_SVGA_Line_Width = 1190;
Default_UXGA_Line_Width = 1922;

When dummy pixel is inserted, the line width changes and frame rate also changes.

SVGA_Line_Width = Default_SVGA_Line_Width + Dummy_pixel
UXGA_Line_Width = Default_UXGA_Line_Width + Dummy_pixel

14.4 Gain

Gain is stored in reg0x00 and Reg0x45[7:6] of bank 1. If only use the gain of Reg0x00, maximum gain of 32x could be reached. It is enough for camera phone. So we don't discuss reg0x45 here.

Gain = (((reg0x00 & 0xf0)>>4) + 1)*(1 + (reg0x00 & 0x0f)/16)

14.5 Banding Filter

14.5.1 Preview

Automatic Banding filter is used for preview.

14.5.2 Capture

Manual banding filter is used for capture. The banding filter calculation is

For 50Hz, the banding filter calculation is

Banding_Filter = Capture_PCLK_Frequency /100 /capture_line_width

For 60Hz, the banding filter calculation is

Banding_Filter = Capture_PCLK_Frequency /120 /capture_line_width

So Capture_Exposure = n*Banding_Filter

n is an integer.

14.6 Auto frame rate

Auto frame rate could be enabled by turn on night mode. When night mode is enabled, the extra line are adjusted automatically.

14.7 Capture Sequence

14.7.1 Preview

// Initialize OV2640 for preview

// Dummy pixel and Dummy line could be inserted for preview
Preview_dummy_pixel =
Preview_dummy_line =

Reg0x2b = Preview_dummy_pixel_reg & 0x00ff;
Reg0x2a = SCCB_read(0x2a);
Reg0x2a = Reg0x2a & 0x0f | (Preview_dummy_pixel_reg & 0x0f00)>>4
SCCB_write(0x2a, Reg0x2a);
SCCB_write(0x2b, Reg0x2b);

```
// update dummy line
Reg0x46 = Preview_dummy_line & 0x00ff;
Reg0x47 = Preview_dummy_line >>8;
SCCB_write(0x46, Reg0x46);
SCCB_write(0x47, Reg0x47);
```

14.7.2 Stop Preview

```
//Stop AE/AG
reg0x13 = SCCB_read(0x13);
Reg0x13 = Reg0x13 & 0xfa;
SCCB_write(0x13, reg0x13);

//Read back preview shutter
reg0x45 = SCCB_read(0x45);
reg0x10 = SCCB_read(0x10);
reg0x04 = SCCB_read(0x04);

Shutter = (reg0x45 & 0x3f)<<10 + reg0x10<<2 + (reg0x04 & 0x03);

//Read back extra line
reg0x2d = SCCB_read(0x2d);
reg0x2e = SCCB_read(0x2e);
Extra_lines = reg0x2d + (reg0x2e<<8);

Preview_Exposure = Shutter + Extra_lines;

//Read Back Gain for preview
reg0x00 = SCCB_read(0x00);
Preview_Gain16 = (((Reg0x00 & 0xf0)>>4) + 1) * (16 + reg0x00 & 0x0f);

//Read back dummy pixels
reg0x2a = SCCB_read(0x2a);
reg0x2b = SCCB_read(0x2b);
Preview_dummy_pixels = (reg0x2a & 0xf0)<<4 + reg0x2b;
```

14.7.3 Calculate Capture Exposure

```
// Dummy Pixel and Dummy Line could be inserted for capture
Capture_dummy_pixel =
Capture_dummy_line =
Preview_PCLK_frequency =
Capture_PCLK_frequency =

// Capture maximum gain could be defined.
// Capture_max_gain16 = capture_max_gain * 16
Capture_max_gain16 =
Preview_line_width = Default_SVGA_Line_Width + Preview_dummy_pixel;
```

```
If (resolution ==SVGA) {
        Capture_line_width = Default_SVGA_Line_Width + capture_Dummy_pixel;
}
else {
        Capture_line_width = Default_UXGA_Line_Width + capture_Dummy_pixel;
}

If (resolution ==SVGA) {
        Capture_maximum_shutter = Default_SVGA_maximum_shutter + capture_dummy_lines;
}
else {
        Capture_maximum_shutter = Default_UXGA_maximum_shutter + capture_dummy_line;
}

Capture_Exposure =
        Preview_Exposure * 2 * Capture_PCLK_Frequency/Preview_PCLK_Frequency *
        Preview_Line_width/Capture_Line_Width;

//Calculate banding filter value
If (50Hz) {
        If (format == RGB) {//RGB indicates raw RGB
                Capture_banding_Filter = Capture_PCLK_Frequency /100 /capture_line_width;
        }
        else {
                Capture_banding_Filter = Capture_PCLK_Frequency/ 100/ (2*capture_line_width);
        }
else {
        If (format == RGB) {
                Capture_banding_Filter = Capture_PCLK_Frequency /120 /capture_line_width;
        }
        else {
                Capture_banding_Filter = Capture_PCLK_frequency /120 /(2*capture_line_width);
        }
}

//redistribute gain and exposure
Gain_Exposure = Preview_Gain16 * Capture_Exposure;
If (Gain_Exposure < Capture_Banding_Filter * 16) {
        // Exposure < 1/100
        Capture_Exposure = Gain_Exposure /16;
        Capture_Gain16 = (Gain_Exposure*2 + 1)/Capture_Exposure/2;
}
else {
        If (Gain_Exposure >  Capture_Maximum_Shutter * 16) {
                // Exposure > Capture_Maximum_Shutter
                Capture_Exposure = Capture_Maximum_Shutter;
                Capture_Gain16 = (Gain_Exposure*2 + 1)/Capture_Maximum_Shutter/2;
```

```
                If (Capture_Gain16 > Capture_Max_Gain16) {
                        // gain reach maximum, insert extra line
                        Capture_Exposure = Gain_Exposure*1.1/Capture_Max_Gain16;
                        // Exposure = n/100
                        Capture_Exposure = Gain_Exposure/16/Capture_banding_filter;
                        Capture_Exposure =
                                Capture_Exposure * Capture_banding_filter;
                        Capture_Gain16 = (Gain_Exposure *2+1)/ Capture_Exposure/2;
                }
        }
        else {
                // 1/100 < Exposure < Capture_Maximum_Shutter, Exposure = n/100
                Capture_Exposure = Gain_Exposure/16/Capture_banding_filter;
                Capture_Exposure =  Capture_Exposure * Capture_banding_filter;
                Capture_Gain16 = (Gain_Exposure*2 +1) / Capture_Exposure/2;
        }
}
```

14.7.4 Switch to UXGA

```
//  Write registers, change to UXGA resolution.
```

14.7.5 Write Registers

```
//write dummy pixels

Reg0x2b = Capture_dummy_pixel_reg & 0x00ff;
Reg0x2a = SCCB_read(0x2a);
Reg0x2a = (Reg0x2a & 0x0f) | ((Capture_dummy_pixel_reg & 0x0f00)>>4);
SCCB_write(0x2a, Reg0x2a);
SCCB_write(0x2b, Reg0x2b);

//Write Dummy Lines
Reg0x46 = Capture_Dummy_lines & 0x00ff;
Reg0x47 = Capture_Dummy_lines>>8;
SCCB_write(0x46, Reg0x46);
SCCB_write(0x47, Reg0x47);

//Write Exposure
If (Capture_Exposure > Capture_maximum_shutter) {
        Shutter = Capture_maximum_shutter;
        Extra_lines = Capture_Exposure – Capture_maximum_shutter;
}
else {
        Shutter = Capture_Exposure;
        Extra_lines = 0;
}
Reg0x04 = SCCB_read(0x04);
Reg0x04 = Reg0x04 & 0xfc | (Shutter & 0x000003);
```

```
Reg0x10 = (Shutter >>2) & 0x00ff;
Reg0x45 = SCCB_read(0x45);
Reg0x45 = (reg0x45 & 0xc0) | ((Shutter >>10) & 0x3f);

SCCB_write(0x45, Reg0x45);
SCCB_write(0x10, Reg0x10);
SCCB_write(0x04, Reg0x04);

// Write extra line
Reg0x2d = Extra_lines & 0x00ff;
Reg0x2e = Extra_lines >> 8;
SCCB_write(0x2d, Reg0x2d);
SCCB_write(0x2e, Reg0x2e);

// Write Gain
Gain = 0;
If (Capture_Gain16 > 16) {
        Capture_Gain16 = Capture_Gain /2;
        Gain = 0x10;
}
If (Capture_Gain16 > 16) {
        Capture_Gain16 = Capture_Gain /2;
        Gain = Gain | 0x20;
}
If (Capture_Gain16 > 16) {
        Capture_Gain16 = Capture_Gain /2;
        Gain = Gain | 0x40;
}
If (Capture_Gain16 > 16) {
        Capture_Gain16 = Capture_Gain /2;
        Gain = Gain | 0x80;
}

Gain = Gain | (Capture_Gain16 -16);

SCCB_write(0x00, Gain);
```

14.7.6 Capture

// Wait for 2 Vsync
// Capture the 3rd frame.

14.7.7 Back to preview

//Write Registers, Change to SVGA

...

//Start AG/AE

```
Reg0x13 = SCCB_read(0x13);
Reg0x13 = Reg0x13 | 0x05;
SCCB_Write(0xff, 0x01);
SCCB_Write(0x13, Reg0x13);
```