OV9650/OV9653 Camera Module Software Application Notes

Last Modified: Dec 12th, 2007

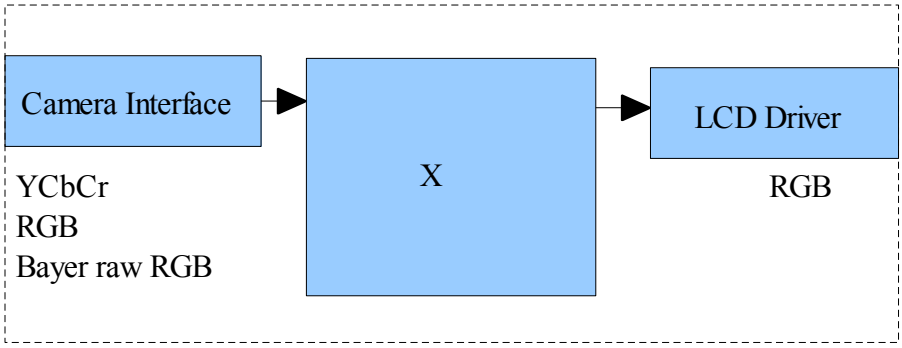
Document Revision: 1.03

Table of Contents

1. How to Select Output format?

OV9650/OV9653 support 3 output format: YcbCr422, RGB565 and Bayer raw RGB , 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 Compression. The Compression 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.

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 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 compression.
c. Sensor output RGB565 for preview, output YCbCr for capture (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 OV9650/OV9653 outputs only VGA format for preview and SXGA for capture. ISP scaled VGA image to other resolution that mobile device needed for LCD display. And the ISP scaled SXGA

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 OV9650/OV9653 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, OV9650/OV9653 output small resolution for preview, and several other resolution for capture. The resolution for capture may include: QQVGA, QVGA, QCIF, CIF, VGA, VGA,SXGA.

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 VGA Preview, 30fps, 24Mhz input clock

```
SCCB_salve_Address = 0x60;
write_SCCB(0x11, 0x80);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x50);
write_SCCB(0x38, 0x92);
write_SCCB(0x35, 0x81);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x6a, 0x7d);
```

3.2 VGA Preview, 15fps, 24 Mhz input clock

```
SCCB_salve_Address = 0x60;
write_SCCB(0x11, 0x81);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x43);
write_SCCB(0x38, 0x12);
write_SCCB(0x35, 0x91);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x6a, 0x3e);
```

3.3 VGA Preview, 25fps, 24 Mhz input clock

```
SCCB_salve_Address = 0x60;
```

write_SCCB(0x11, 0x80);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x50);
write_SCCB(0x38, 0x92);
write_SCCB(0x35, 0x81);
write_SCCB(0x2a, 0x10);
write_SCCB(0x2b, 0x40);
write_SCCB(0x6a, 0x7d);

3.4 VGA Preview, 12.5fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0x11, 0x81);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x43);
write_SCCB(0x38, 0x12);
write_SCCB(0x35, 0x91);
write_SCCB(0x2a, 0x10);
write_SCCB(0x2b, 0x40);
write_SCCB(0x6a, 0x3e);

3.5 SXGA Capture, 7.5fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0x11, 0x80);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x43);
write_SCCB(0x38, 0x12);
write_SCCB(0x35, 0x91);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x6a, 0x41);

3.6 SXGA Capture, 6.25fps, 24 Mhz input clock

SCCB_salve_Address = 0x60;
write_SCCB(0x11, 0x80);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x43);
write_SCCB(0x38, 0x12);
write_SCCB(0x35, 0x91);
write_SCCB(0x2a, 0x10);
write_SCCB(0x2b, 0x34);
write_SCCB(0x6a, 0x41);

4. How to set Night Mode Preview

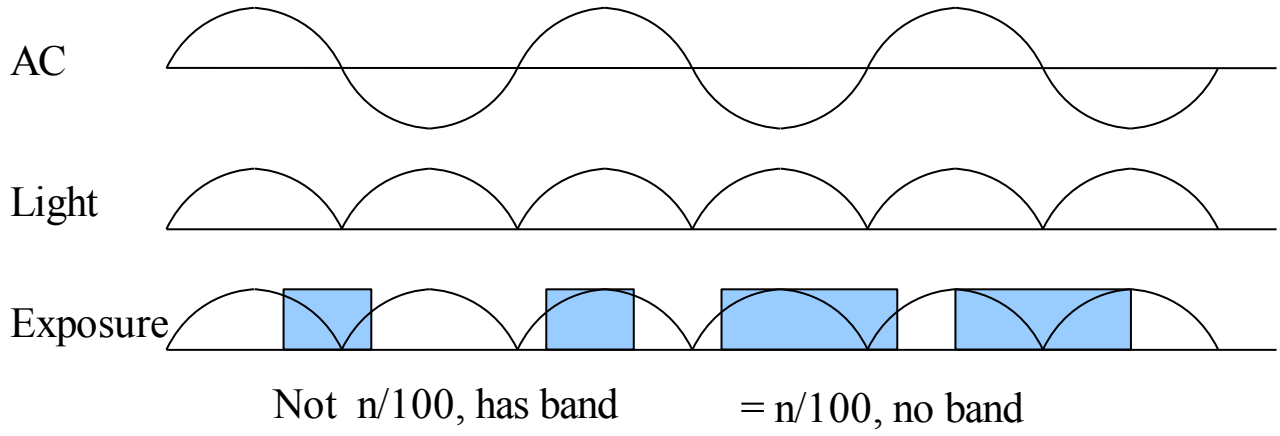
3.75fps night mode for 60Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;

write_SCCB(0x11, 0x87);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x43);
write_SCCB(0x38, 0x12);
write_SCCB(0x35, 0x91);
write_SCCB(0x2a, 0x00);
write_SCCB(0x2b, 0x00);
write_SCCB(0x6a, 0x10);

3.125fps night mode for 50Hz light environment, 24Mhz clock input
SCCB_salve_Address = 0x60;
write_SCCB(0x11, 0x87);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x39, 0x43);
write_SCCB(0x38, 0x12);
write_SCCB(0x35, 0x91);
write_SCCB(0x2a, 0x10);
write_SCCB(0x2b, 0x40);
write_SCCB(0x6a, 0x10);

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 OV9650/OV9653 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 VGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0x13, 0xe7); //banding filter enable
write_SCCB(0x2a, 0x00);//Select 60Hz banding filter
write_SCCB(0x2b, 0x00);//Select 60Hz banding filter
write_SCCB(0x6a, 0x7d);//banding filter

banding filter setting for 15fps VGA preview, 24Mhz input clock
write_SCCB(0x13, 0xe7); //banding filter enable
write_SCCB(0x2a, 0x00);//Select 60Hz banding filter
write_SCCB(0x2b, 0x00);//Select 60Hz banding filter
write_SCCB(0x6a, 0x3e);//banding filter

banding filter setting for 25fps VGA preview, 24Mhz input clock
write_SCCB(0x13, 0xe7); //banding filter enable
write_SCCB(0x2a, 0x10);//Select 50Hz banding filter

write_SCCB(0x2b, 0x40);//Select 50Hz banding filter
write_SCCB(0x6a, 0x7d);//banding filter

banding filter setting for 12.5fps VGA preview, 24Mhz input clock
SCCB_salve_Address = 0x60;
write_SCCB(0x13, 0xe7); //banding filter enable
write_SCCB(0x2a, 0x10);//Select 50Hz banding filter
write_SCCB(0x2b, 0x40);//Select 50Hz banding filter
write_SCCB(0x6a, 0x3e);//banding filter

5.4 Remove Light Band in Capture

Please See Capture Sequence

5.5When 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 OV9650/OV9653. So there is no way to remove light band in this condition.

6. White Balance

OV9650/OV9653 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(0x60, 0x8c); // 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 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

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.

OV9650/OV9653 has built-in white pixel correction function. If OV9650/OV9653 output YCbCr, RGB565, the white pixel correction function could be enabled to fix white 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 OV9650/OV9653

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 OV9650/OV9653. It should always be turned on.

9. Video Mode

Video mode need high frame rate, usually fixed 15fps for 60Hz, and 14.3 for 50Hz. There is no night mode for video mode.

10. Digital zoom

If OV9650/OV9653 output image smaller than VGA, it may support continuous digital zoom. For example

SXGA	no digital zoom supported
VGA	1x, 2x
QVGA	1x, 4x
QQVGA	1x, 2x, 4x, 8x

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

11. OV9650/OV9653 Functions

11.1 Light Mode

Auto
```
SCCB_salve_Address = 0x60;
write_SCCB(0x13 ,0xe7);
write_SCCB(0x11 ,0x81);
write_SCCB(0x41 ,0x02);
write_SCCB(0x3f ,0xa6);
write_SCCB(0x3d ,0x92);
write_SCCB(0x66 ,0x01);
write_SCCB(0x14 ,0x1e);
```

Sunny
```
SCCB_salve_Address = 0x60;
write_SCCB(0x13 ,0xc5);
write_SCCB(0x01 ,0x74);
write_SCCB(0x02 ,0x90);
write_SCCB(0x14 ,0x0e);
write_SCCB(0x11 ,0x81);
write_SCCB(0x41 ,0x02);
write_SCCB(0x3f ,0xa6);
write_SCCB(0x3d ,0x92);
write_SCCB(0x66 ,0x01);
```

Cloudy
```
SCCB_salve_Address = 0x60;
```

```
write_SCCB(0x13 ,0xc5);
write_SCCB(0x01 ,0x74);
write_SCCB(0x02 ,0x90);
write_SCCB(0x14 ,0x1e);
write_SCCB(0x11 ,0x81);
write_SCCB(0x41 ,0x02);
write_SCCB(0x3f ,0xa6);
write_SCCB(0x3d ,0x92);
write_SCCB(0x66 ,0x01);
```

Office

```
SCCB_salve_Address = 0x60;
write_SCCB(0x13 ,0xe5);
write_SCCB(0x01 ,0x97);
write_SCCB(0x02 ,0x7d);
write_SCCB(0x14 ,0x1e);
write_SCCB(0x11 ,0x81);
write_SCCB(0x41 ,0x02);
write_SCCB(0x3f ,0xa6);
write_SCCB(0x3d ,0x92);
write_SCCB(0x66 ,0x01);
```

Home

```
SCCB_salve_Address = 0x60;
write_SCCB(0x13 ,0xe5);
write_SCCB(0x01 ,0x9c);
write_SCCB(0x02 ,0x54);
write_SCCB(0x14 ,0x1e);
write_SCCB(0x11 ,0x81);
write_SCCB(0x41 ,0x02);
write_SCCB(0x3f ,0xa6);
write_SCCB(0x3d ,0x92);
write_SCCB(0x66 ,0x01);
```

Night

```
SCCB_salve_Address = 0x60;
write_SCCB(0x11 ,0x83);
write_SCCB(0x41 ,0x00);
write_SCCB(0x3f ,0xf3);
write_SCCB(0x3d ,0x12);
write_SCCB(0x66 ,0x00);
write_SCCB(0x13 ,0xe7);
write_SCCB(0x14 ,0x1e);
```

11.2 Color Saturation

The color saturation of OV9650/OV9653 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(0x4f ,0x57);
write_SCCB(0x50 ,0x5c);
write_SCCB(0x51 ,0x05);
write_SCCB(0x52 ,0x1b);
write_SCCB(0x53 ,0x39);
write_SCCB(0x54 ,0x54);

Saturation + 1
SCCB_salve_Address = 0x60;
write_SCCB(0x4f ,0x46);
write_SCCB(0x50 ,0x49);
write_SCCB(0x51 ,0x04);
write_SCCB(0x52 ,0x16);
write_SCCB(0x53 ,0x2e);
write_SCCB(0x54 ,0x43);

Saturation 0
SCCB_salve_Address = 0x60;
write_SCCB(0x4f ,0x3a);
write_SCCB(0x50 ,0x3d);
write_SCCB(0x51 ,0x03);
write_SCCB(0x52 ,0x12);
write_SCCB(0x53 ,0x26);
write_SCCB(0x54 ,0x38);

Saturation -1
SCCB_salve_Address = 0x60;
write_SCCB(0x4f ,0x2e);
write_SCCB(0x50 ,0x31);
write_SCCB(0x51 ,0x02);
write_SCCB(0x52 ,0x0e);
write_SCCB(0x53 ,0x1E);
write_SCCB(0x54 ,0x2d);

Saturation - 2
SCCB_salve_Address = 0x60;
write_SCCB(0x4f ,0x1d);
write_SCCB(0x50 ,0x1f);
write_SCCB(0x51 ,0x02);
write_SCCB(0x52 ,0x09);
write_SCCB(0x53 ,0x13);
write_SCCB(0x54 ,0x1c);

11.3 Brightness

The brightness of OV9650/OV9653 could be adjusted. Higher brightness will make the picture more bright. The side effect of higher brightness is the picture looks frogy.

Brightness +3
SCCB_salve_Address = 0x60;
write_SCCB(0x24 ,0xd8);
write_SCCB(0x25 ,0xd0);
write_SCCB(0x26 ,0xfa);

Brightness +2
SCCB_salve_Address = 0x60;
write_SCCB(0x24 ,0xc4);
write_SCCB(0x25 ,0xbf);
write_SCCB(0x26 ,0xe9);

Brightness +1
SCCB_salve_Address = 0x60;
write_SCCB(0x24 ,0x90);
write_SCCB(0x25 ,0x84);
write_SCCB(0x26 ,0xd4);

Brightness 0
SCCB_salve_Address = 0x60;
write_SCCB(0x24 ,0x70);
write_SCCB(0x25 ,0x64);
write_SCCB(0x26 ,0xc3);

Brightness -1
SCCB_salve_Address = 0x60;
write_SCCB(0x24 ,0x50);
write_SCCB(0x25 ,0x44);
write_SCCB(0x26 ,0x92);

Brightness -2
SCCB_salve_Address = 0x60;
write_SCCB(0x24 ,0x3d);
write_SCCB(0x25 ,0x30);
write_SCCB(0x26 ,0x71);

Brightness -3
SCCB_salve_Address = 0x60;
write_SCCB(0x24 ,0x1c);
write_SCCB(0x25 ,0x12);
write_SCCB(0x26 ,0x50);

11.4 Contrast

The contrast of OV9650/OV9653 could be adjusted. Higher contrast will make the picture sharp. But the side effect is loosing dynamic range.

Contrast +3
SCCB_salve_Address = 0x60;
write_SCCB(0x7c ,0x04);
write_SCCB(0x7d ,0x09);
write_SCCB(0x7e ,0x14);
write_SCCB(0x7f ,0x28);
write_SCCB(0x80 ,0x32);
write_SCCB(0x81 ,0x3C);
write_SCCB(0x82 ,0x46);
write_SCCB(0x83 ,0x4F);
write_SCCB(0x84 ,0x58);
write_SCCB(0x85 ,0x61);
write_SCCB(0x86 ,0x73);
write_SCCB(0x87 ,0x85);
write_SCCB(0x88 ,0xA5);
write_SCCB(0x89 ,0xc5);
write_SCCB(0x8a ,0xDD);
write_SCCB(0x6c ,0x40);
write_SCCB(0x6d ,0x50);
write_SCCB(0x6e ,0x58);
write_SCCB(0x6f ,0x50);
write_SCCB(0x70 ,0x50);
write_SCCB(0x71 ,0x50);
write_SCCB(0x72 ,0x50);
write_SCCB(0x73 ,0x48);
write_SCCB(0x74 ,0x48);
write_SCCB(0x75 ,0x48);
write_SCCB(0x76 ,0x48);
write_SCCB(0x77 ,0x48);
write_SCCB(0x78 ,0x40);
write_SCCB(0x79 ,0x40);
write_SCCB(0x7a ,0x30);
write_SCCB(0x7b ,0x2E);

Contrast +2
SCCB_salve_Address = 0x60;
write_SCCB(0x7c ,0x05);
write_SCCB(0x7d ,0x0b);
write_SCCB(0x7e ,0x16);
write_SCCB(0x7f ,0x2c);
write_SCCB(0x80 ,0x37);
write_SCCB(0x81 ,0x41);
write_SCCB(0x82 ,0x4b);

```
write_SCCB(0x83 ,0x55);
write_SCCB(0x84 ,0x5f);
write_SCCB(0x85 ,0x69);
write_SCCB(0x86 ,0x7c);
write_SCCB(0x87 ,0x8f);
write_SCCB(0x88 ,0xb1);
write_SCCB(0x89 ,0xcf);
write_SCCB(0x8a ,0xe5);
write_SCCB(0x6c ,0x50);
write_SCCB(0x6d ,0x60);
write_SCCB(0x6e ,0x58);
write_SCCB(0x6f ,0x58);
write_SCCB(0x70 ,0x58);
write_SCCB(0x71 ,0x50);
write_SCCB(0x72 ,0x50);
write_SCCB(0x73 ,0x50);
write_SCCB(0x74 ,0x50);
write_SCCB(0x75 ,0x50);
write_SCCB(0x76 ,0x4c);
write_SCCB(0x77 ,0x4c);
write_SCCB(0x78 ,0x44);
write_SCCB(0x79 ,0x3c);
write_SCCB(0x7a ,0x2c);
write_SCCB(0x7b ,0x23);
```

Contrast +1

```
SCCB_salve_Address = 0x60;
write_SCCB(0x7C ,0x04);
write_SCCB(0x7D ,0x09);
write_SCCB(0x7E ,0x13);
write_SCCB(0x7F ,0x29);
write_SCCB(0x80 ,0x35);
write_SCCB(0x81 ,0x41);
write_SCCB(0x82 ,0x4D);
write_SCCB(0x83 ,0x59);
write_SCCB(0x84 ,0x64);
write_SCCB(0x85 ,0x6F);
write_SCCB(0x86 ,0x85);
write_SCCB(0x87 ,0x97);
write_SCCB(0x88 ,0xB7);
write_SCCB(0x89 ,0xCF);
write_SCCB(0x8A ,0xE3);
write_SCCB(0x6C ,0x40);
write_SCCB(0x6D ,0x50);
write_SCCB(0x6E ,0x50);
write_SCCB(0x6F ,0x58);
write_SCCB(0x70 ,0x60);
write_SCCB(0x71 ,0x60);
```

```
write_SCCB(0x72 ,0x60);
write_SCCB(0x73 ,0x60);
write_SCCB(0x74 ,0x58);
write_SCCB(0x75 ,0x58);
write_SCCB(0x76 ,0x58);
write_SCCB(0x77 ,0x48);
write_SCCB(0x78 ,0x40);
write_SCCB(0x79 ,0x30);
write_SCCB(0x7A ,0x28);
write_SCCB(0x7B ,0x26);
```

Contrast 0

```
SCCB_salve_Address = 0x60;
write_SCCB(0x7C ,0x04);
write_SCCB(0x7D ,0x07);
write_SCCB(0x7E ,0x10);
write_SCCB(0x7F ,0x28);
write_SCCB(0x80 ,0x36);
write_SCCB(0x81 ,0x44);
write_SCCB(0x82 ,0x52);
write_SCCB(0x83 ,0x60);
write_SCCB(0x84 ,0x6C);
write_SCCB(0x85 ,0x78);
write_SCCB(0x86 ,0x8C);
write_SCCB(0x87 ,0x9E);
write_SCCB(0x88 ,0xBB);
write_SCCB(0x89 ,0xD2);
write_SCCB(0x8A ,0xE6);
write_SCCB(0x6C ,0x40);
write_SCCB(0x6D ,0x30);
write_SCCB(0x6E ,0x48);
write_SCCB(0x6F ,0x60);
write_SCCB(0x70 ,0x70);
write_SCCB(0x71 ,0x70);
write_SCCB(0x72 ,0x70);
write_SCCB(0x73 ,0x70);
write_SCCB(0x74 ,0x60);
write_SCCB(0x75 ,0x60);
write_SCCB(0x76 ,0x50);
write_SCCB(0x77 ,0x48);
write_SCCB(0x78 ,0x3A);
write_SCCB(0x79 ,0x2E);
write_SCCB(0x7A ,0x28);
write_SCCB(0x7B ,0x22);
```

Contrast -1

```
SCCB_salve_Address = 0x60;
write_SCCB(0x7C ,0x02);
```

```
write_SCCB(0x7D ,0x06);
write_SCCB(0x7E ,0x16);
write_SCCB(0x7F ,0x3A);
write_SCCB(0x80 ,0x4C);
write_SCCB(0x81 ,0x5C);
write_SCCB(0x82 ,0x6A);
write_SCCB(0x83 ,0x78);
write_SCCB(0x84 ,0x84);
write_SCCB(0x85 ,0x8E);
write_SCCB(0x86 ,0x9E);
write_SCCB(0x87 ,0xAE);
write_SCCB(0x88 ,0xC6);
write_SCCB(0x89 ,0xDA);
write_SCCB(0x8A ,0xEA);
write_SCCB(0x6C ,0x20);
write_SCCB(0x6D ,0x40);
write_SCCB(0x6E ,0x80);
write_SCCB(0x6F ,0x90);
write_SCCB(0x70 ,0x90);
write_SCCB(0x71 ,0x80);
write_SCCB(0x72 ,0x70);
write_SCCB(0x73 ,0x70);
write_SCCB(0x74 ,0x60);
write_SCCB(0x75 ,0x50);
write_SCCB(0x76 ,0x40);
write_SCCB(0x77 ,0x40);
write_SCCB(0x78 ,0x30);
write_SCCB(0x79 ,0x28);
write_SCCB(0x7A ,0x20);
write_SCCB(0x7B ,0x1c);
```

Contrast -2

```
SCCB_salve_Address = 0x60;
write_SCCB(0x7C ,0x02);
write_SCCB(0x7D ,0x07);
write_SCCB(0x7E ,0x1f);
write_SCCB(0x7F ,0x49);
write_SCCB(0x80 ,0x5a);
write_SCCB(0x81 ,0x6a);
write_SCCB(0x82 ,0x79);
write_SCCB(0x83 ,0x87);
write_SCCB(0x84 ,0x94);
write_SCCB(0x85 ,0x9f);
write_SCCB(0x86 ,0xaf);
write_SCCB(0x87 ,0xbb);
write_SCCB(0x88 ,0xcf);
write_SCCB(0x89 ,0xdf);
write_SCCB(0x8A ,0xee);
```

```
write_SCCB(0x6C ,0x20);
write_SCCB(0x6D ,0x50);
write_SCCB(0x6E ,0xc0);
write_SCCB(0x6F ,0xa8);
write_SCCB(0x70 ,0x88);
write_SCCB(0x71 ,0x80);
write_SCCB(0x72 ,0x78);
write_SCCB(0x73 ,0x70);
write_SCCB(0x74 ,0x68);
write_SCCB(0x75 ,0x58);
write_SCCB(0x76 ,0x40);
write_SCCB(0x77 ,0x30);
write_SCCB(0x78 ,0x28);
write_SCCB(0x79 ,0x20);
write_SCCB(0x7A ,0x1e);
write_SCCB(0x7B ,0x17);
```

Contrast -3

```
SCCB_salve_Address = 0x60;
write_SCCB(0x6C ,0x20);
write_SCCB(0x6D ,0x50);
write_SCCB(0x6E ,0x80);
write_SCCB(0x6F ,0xC0);
write_SCCB(0x70 ,0xC0);
write_SCCB(0x71 ,0xA0);
write_SCCB(0x72 ,0x90);
write_SCCB(0x73 ,0x78);
write_SCCB(0x74 ,0x78);
write_SCCB(0x75 ,0x78);
write_SCCB(0x76 ,0x40);
write_SCCB(0x77 ,0x20);
write_SCCB(0x78 ,0x20);
write_SCCB(0x79 ,0x20);
write_SCCB(0x7A ,0x14);
write_SCCB(0x7B ,0x0E);
write_SCCB(0x7C ,0x02);
write_SCCB(0x7D ,0x07);
write_SCCB(0x7E ,0x17);
write_SCCB(0x7F ,0x47);
write_SCCB(0x80 ,0x5F);
write_SCCB(0x81 ,0x73);
write_SCCB(0x82 ,0x85);
write_SCCB(0x83 ,0x94);
write_SCCB(0x84 ,0xA3);
write_SCCB(0x85 ,0xB2);
write_SCCB(0x86 ,0xC2);
write_SCCB(0x87 ,0xCA);
write_SCCB(0x88 ,0xDA);
```

write_SCCB(0x89 ,0xEA);
write_SCCB(0x8A ,0xF4);

11.5 Special effects

OV9650/OV9653 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.

Antique
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x11);
write_SCCB(0x67, 0xa0);
write_SCCB(0x68, 0x40);

Bluish
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x11);
write_SCCB(0x67, 0x80);
write_SCCB(0x68, 0xc0);

Greenish
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x11);
write_SCCB(0x67, 0x40);
write_SCCB(0x68, 0x40);

Redish
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x11);
write_SCCB(0x67, 0xc0);
write_SCCB(0x68, 0x80);

B&W
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x11);
write_SCCB(0x67, 0xc80);
write_SCCB(0x68, 0x80);

Negative
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x21);
write_SCCB(0x67, 0x80);
write_SCCB(0x68, 0x80);

B&W negative
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x31);

write_SCCB(0x67, 0x80);
write_SCCB(0x68, 0x80);

Normal
SCCB_salve_Address = 0x60;
write_SCCB(0x3a, 0x01);
write_SCCB(0x67, 0xc0);
write_SCCB(0x68, 0x80);

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 lens. The lens shading correction function of OV9650/OV9653 could be turned on to compensate the corner brightness and make the whole picture looks same bright.

12.2 Lens Correction

The lens correction settings should be adjusted by OmniVision FAE in optical lab. To get best lens correction setting which could cover mass production modules, at least 20 camera module samples are required. If the camera module samples used to adjust lens correction are too few, the adjusted setting may not be able to cover all camera modules.

Please contact with OmniVison local FAE for lens correction adjustment.

12.3 2 Module Supplier with Different Lens

The lens correction of OV9650/OV9653 should be adjusted for every sensor-lens combination. The setting could cover only one lens. The same setting could not cover 2 lens or more. For example, if a project has 2 module suppliers, then the 2 module suppliers should use same lens. If the 2 module suppliers uses different lens for same project, then the single lens correction setting can not cover both camera modules. If the lens correction is good for lens A, then it is not good for lens B. From sensor point of view, sensor can not detect which lens is using.

This issue could be solved by adding module code for the modules with different lens. A simple way to separate module with lens A from module with lens B is add pull up/pull down registers on an NC pin of camera module. For example, module A add a pull up resistor on the NC pin, module add a pull down resistor on the same NC pin. Then mobile phone could detect the status of the NC pin to check which lens is using. Then apply different lens correction setting for different lens.

12.4 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.

12.5 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 OV9650/OV9653 the focus distance is about 90~110cm. The depth of field is about from 40~55cm to infinite. If checking resolution of camera module, the resolution chart should be placed 90~110 cm away from camera module.

12.6 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.

12.7 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 strong lens flare.

13. Reference Settings

13.1 RAW Setting

```
//SXGA 15fps
//
write_SCCB(0x12, 0x80);
write_SCCB(0x11, 0x81);
write_SCCB(0x6b, 0x4a);
write_SCCB(0x3b, 0x01);
write_SCCB(0x6a, 0x83);
write_SCCB(0x13, 0xe2);
write_SCCB(0x10, 0x00);
write_SCCB(0x00, 0x00);
write_SCCB(0x01, 0x80);
write_SCCB(0x02, 0x80);
write_SCCB(0x13, 0xe7);
//
write_SCCB(0x39, 0x50);
write_SCCB(0x38, 0x93);
write_SCCB(0x37, 0x00);
write_SCCB(0x35, 0x81);
write_SCCB(0x0e, 0xa0);
//
write_SCCB(0xa8, 0x80);
write_SCCB(0x12, 0x05);
write_SCCB(0x04, 0x00);
write_SCCB(0x0c, 0x00);
```

```
write_SCCB(0x0d, 0x00);
write_SCCB(0x18, 0xbb);
write_SCCB(0x17, 0x1b);
write_SCCB(0x32, 0xa4);
write_SCCB(0x19, 0x01);
write_SCCB(0x1a, 0x81);
write_SCCB(0x03, 0x12);
//
write_SCCB(0x1b, 0x00);
write_SCCB(0x16, 0x07);
write_SCCB(0x33, 0xe2); //c0 for internal regulator
write_SCCB(0x34, 0xbf);
write_SCCB(0x41, 0x00);
write_SCCB(0x96, 0x04);
//
write_SCCB(0x3d, 0x19);
write_SCCB(0x69, 0x40);
write_SCCB(0x3a, 0x0d);
write_SCCB(0x8e, 0x00);
//
write_SCCB(0x3c, 0x73);
write_SCCB(0x8f, 0xdf);
write_SCCB(0x8b, 0x06);
write_SCCB(0x8c, 0x20);
write_SCCB(0x94, 0x88);
write_SCCB(0x95, 0x88);
write_SCCB(0x40, 0xc1);
write_SCCB(0x29, 0x3f); //2f for internal regulator
write_SCCB(0x0f, 0x42);
write_SCCB(0xa5, 0x80);
write_SCCB(0x1e, 0x04);
write_SCCB(0xa9, 0xb8);
write_SCCB(0xaa, 0x92);
write_SCCB(0xab, 0x0a);
//
write_SCCB(0x24, 0x68);
write_SCCB(0x25, 0x5c);
write_SCCB(0x26, 0xc3);
write_SCCB(0x14, 0x2e);
```

13.2 RGB565 Setting

13.2.1 VGA mode

```
write_SCCB(0x12, 0x80);
write_SCCB(0x11, 0x83);
write_SCCB(0x6b, 0x4a);
write_SCCB(0x6a, 0x3e);
write_SCCB(0x3b, 0x09);
write_SCCB(0x13, 0xe0);
write_SCCB(0x01, 0x80);
write_SCCB(0x02, 0x80);
write_SCCB(0x00, 0x00);
write_SCCB(0x10, 0x00);
write_SCCB(0x13, 0xe5);
```

```
//
write_SCCB(0x39, 0x43); //50 for 30fps
write_SCCB(0x38, 0x12); //92 for 30fps
write_SCCB(0x37, 0x00);
write_SCCB(0x35, 0x91); //81 for 30fps
write_SCCB(0x0e, 0xa0);
write_SCCB(0x1e, 0x04);
//
write_SCCB(0xa8, 0x80);
write_SCCB(0x12, 0x44);
write_SCCB(0x04, 0x00);
write_SCCB(0x0c, 0x04);
write_SCCB(0x0d, 0x80);
write_SCCB(0x18, 0xc6);
write_SCCB(0x17, 0x26);
write_SCCB(0x32, 0xad);
write_SCCB(0x03, 0x00);
write_SCCB(0x1a, 0x3d);
write_SCCB(0x19, 0x01);
write_SCCB(0x3f, 0xa6);
write_SCCB(0x14, 0x2e);
write_SCCB(0x15, 0x02);
write_SCCB(0x41, 0x00);
write_SCCB(0x42, 0x08);
//
write_SCCB(0x1b, 0x00);
write_SCCB(0x16, 0x06);
write_SCCB(0x33, 0xe2); //c0 for internal regulator
write_SCCB(0x34, 0xbf);
write_SCCB(0x96, 0x04);
write_SCCB(0x3a, 0x00);
write_SCCB(0x8e, 0x00);
//
write_SCCB(0x3c, 0x77);
write_SCCB(0x8b, 0x06);
write_SCCB(0x94, 0x88);
write_SCCB(0x95, 0x88);
write_SCCB(0x40, 0xd1);
write_SCCB(0x29, 0x3f); //2f for internal regulator
write_SCCB(0x0f, 0x42);
//
write_SCCB(0x3d, 0x90);
write_SCCB(0x69, 0x40);
write_SCCB(0x5c, 0xb9);
write_SCCB(0x5d, 0x96);
write_SCCB(0x5e, 0x10);
write_SCCB(0x59, 0xc0);
write_SCCB(0x5a, 0xaf);
write_SCCB(0x5b, 0x55);
write_SCCB(0x43, 0xf0);
write_SCCB(0x44, 0x10);
write_SCCB(0x45, 0x68);
write_SCCB(0x46, 0x96);
write_SCCB(0x47, 0x60);
write_SCCB(0x48, 0x80);
write_SCCB(0x5f, 0xe0);
write_SCCB(0x60, 0x8c); //0c for advanced AWB (related to lens)
write_SCCB(0x61, 0x20);
```

```
write_SCCB(0xa5, 0xd9);
write_SCCB(0xa4, 0x74);
write_SCCB(0x8d, 0x02);
write_SCCB(0x13, 0xe7);
//
write_SCCB(0x4f, 0xb7);
write_SCCB(0x50, 0x2e);
write_SCCB(0x51, 0x09);
write_SCCB(0x52, 0x1f);
write_SCCB(0x53, 0xb1);
write_SCCB(0x54, 0x12);
write_SCCB(0x55, 0x06);
write_SCCB(0x56, 0x55);
write_SCCB(0x57, 0xdb);
write_SCCB(0x58, 0x77);
//
write_SCCB(0x8c, 0x23);
write_SCCB(0x3e, 0x02);
write_SCCB(0xa9, 0xb8);
write_SCCB(0xaa, 0x92);
write_SCCB(0xab, 0x0a);
//
write_SCCB(0x8f, 0xdf);
write_SCCB(0x90, 0x00);
write_SCCB(0x91, 0x00);
write_SCCB(0x9f, 0x00);
write_SCCB(0xa0, 0x00);
write_SCCB(0x3a, 0x01);
//
write_SCCB(0x24, 0x70);
write_SCCB(0x25, 0x64);
write_SCCB(0x26, 0xc3);
//
write_SCCB(0x2a, 0x00); //10 for 50Hz
write_SCCB(0x2b, 0x00); //40 for 50Hz
//
//gamma
write_SCCB(0x6c, 0x40);
write_SCCB(0x6d, 0x30);
write_SCCB(0x6e, 0x4b);
write_SCCB(0x6f, 0x60);
write_SCCB(0x70, 0x70);
write_SCCB(0x71, 0x70);
write_SCCB(0x72, 0x70);
write_SCCB(0x73, 0x70);
write_SCCB(0x74, 0x60);
write_SCCB(0x75, 0x60);
write_SCCB(0x76, 0x50);
write_SCCB(0x77, 0x48);
write_SCCB(0x78, 0x3a);
write_SCCB(0x79, 0x2e);
write_SCCB(0x7a, 0x28);
write_SCCB(0x7b, 0x22);
write_SCCB(0x7c, 0x04);
write_SCCB(0x7d, 0x07);
write_SCCB(0x7e, 0x10);
write_SCCB(0x7f, 0x28);
write_SCCB(0x80, 0x36);
```

```
write_SCCB(0x81, 0x44);
write_SCCB(0x82, 0x52);
write_SCCB(0x83, 0x60);
write_SCCB(0x84, 0x6c);
write_SCCB(0x85, 0x78);
write_SCCB(0x86, 0x8c);
write_SCCB(0x87, 0x9e);
write_SCCB(0x88, 0xbb);
write_SCCB(0x89, 0xd2);
write_SCCB(0x8a, 0xe6);
//
write_SCCB(0x09, 0x11);
write_SCCB(0x09, 0x01);
```

13.2.2 SXGA mode

```
write_SCCB(0x12, 0x80);
write_SCCB(0x11, 0x81);
write_SCCB(0x6b, 0x4a);
write_SCCB(0x6a, 0x41);
write_SCCB(0x3b, 0x09);
write_SCCB(0x13, 0xe0);
write_SCCB(0x01, 0x80);
write_SCCB(0x02, 0x80);
write_SCCB(0x00, 0x00);
write_SCCB(0x10, 0x00);
write_SCCB(0x13, 0xe5);
//
write_SCCB(0x39, 0x43); //50 for 15fps
write_SCCB(0x38, 0x12); //93 for 15fps
write_SCCB(0x37, 0x00);
write_SCCB(0x35, 0x91); //81 for 15fps
write_SCCB(0x0e, 0xa0);
write_SCCB(0x1e, 0x04);
//
write_SCCB(0xa8, 0x80);
write_SCCB(0x12, 0x04);
write_SCCB(0x04, 0x00);
write_SCCB(0x0c, 0x00);
write_SCCB(0x0d, 0x00);
write_SCCB(0x18, 0xbd);
write_SCCB(0x17, 0x1d);
write_SCCB(0x32, 0xad);
write_SCCB(0x03, 0x12);
write_SCCB(0x1a, 0x81);
write_SCCB(0x19, 0x01);
write_SCCB(0x14, 0x2e);
write_SCCB(0x15, 0x00);
write_SCCB(0x3f, 0xa6);
write_SCCB(0x41, 0x00);
write_SCCB(0x42, 0x08);
//
write_SCCB(0x1b, 0x00);
write_SCCB(0x16, 0x06);
write_SCCB(0x33, 0xe2); //c0 for internal regulator
write_SCCB(0x34, 0xbf);
write_SCCB(0x96, 0x04);
write_SCCB(0x3a, 0x00);
```

```
write_SCCB(0x8e, 0x00);
//
write_SCCB(0x3c, 0x77);
write_SCCB(0x8b, 0x06);
write_SCCB(0x94, 0x88);
write_SCCB(0x95, 0x88);
write_SCCB(0x40, 0xd1);
write_SCCB(0x29, 0x3f); //2f for internal regulator
write_SCCB(0x0f, 0x42);
//
write_SCCB(0x3d, 0x90);
write_SCCB(0x69, 0x40);
write_SCCB(0x5c, 0xb9);
write_SCCB(0x5d, 0x96);
write_SCCB(0x5e, 0x10);
write_SCCB(0x59, 0xc0);
write_SCCB(0x5a, 0xaf);
write_SCCB(0x5b, 0x55);
write_SCCB(0x43, 0xf0);
write_SCCB(0x44, 0x10);
write_SCCB(0x45, 0x68);
write_SCCB(0x46, 0x96);
write_SCCB(0x47, 0x60);
write_SCCB(0x48, 0x80);
write_SCCB(0x5f, 0xe0);
write_SCCB(0x60, 0x8c); //0c for advanced AWB (Related to lens)
write_SCCB(0x61, 0x20);
write_SCCB(0xa5, 0xd9);
write_SCCB(0xa4, 0x74);
write_SCCB(0x8d, 0x02);
write_SCCB(0x13, 0xe7);
//
write_SCCB(0x4f, 0xb7);
write_SCCB(0x50, 0x2e);
write_SCCB(0x51, 0x09);
write_SCCB(0x52, 0x1f);
write_SCCB(0x53, 0xb1);
write_SCCB(0x54, 0x12);
write_SCCB(0x55, 0x06);
write_SCCB(0x56, 0x55);
write_SCCB(0x57, 0xdb);
write_SCCB(0x58, 0x77);
//
write_SCCB(0x8c, 0x23);
write_SCCB(0x3e, 0x02);
write_SCCB(0xa9, 0xb8);
write_SCCB(0xaa, 0x92);
write_SCCB(0xab, 0x0a);
//
write_SCCB(0x8f, 0xdf);
write_SCCB(0x90, 0x00);
write_SCCB(0x91, 0x00);
write_SCCB(0x9f, 0x00);
write_SCCB(0xa0, 0x00);
write_SCCB(0x3a, 0x01);
//
write_SCCB(0x24, 0x70);
write_SCCB(0x25, 0x64);
```

```
write_SCCB(0x26, 0xc3);
//
write_SCCB(0x2a, 0x00); //10 for 50Hz
write_SCCB(0x2b, 0x00); //34 for 50Hz
//
//gamma
write_SCCB(0x6c, 0x40);
write_SCCB(0x6d, 0x30);
write_SCCB(0x6e, 0x4b);
write_SCCB(0x6f, 0x60);
write_SCCB(0x70, 0x70);
write_SCCB(0x71, 0x70);
write_SCCB(0x72, 0x70);
write_SCCB(0x73, 0x70);
write_SCCB(0x74, 0x60);
write_SCCB(0x75, 0x60);
write_SCCB(0x76, 0x50);
write_SCCB(0x77, 0x48);
write_SCCB(0x78, 0x3a);
write_SCCB(0x79, 0x2e);
write_SCCB(0x7a, 0x28);
write_SCCB(0x7b, 0x22);
write_SCCB(0x7c, 0x04);
write_SCCB(0x7d, 0x07);
write_SCCB(0x7e, 0x10);
write_SCCB(0x7f, 0x28);
write_SCCB(0x80, 0x36);
write_SCCB(0x81, 0x44);
write_SCCB(0x82, 0x52);
write_SCCB(0x83, 0x60);
write_SCCB(0x84, 0x6c);
write_SCCB(0x85, 0x78);
write_SCCB(0x86, 0x8c);
write_SCCB(0x87, 0x9e);
write_SCCB(0x88, 0xbb);
write_SCCB(0x89, 0xd2);
write_SCCB(0x8a, 0xe6);
//
write_SCCB(0x09, 0x11);
write_SCCB(0x09, 0x01);
```

13.3 YUV Setting

13.3.1 VGA

```
write_SCCB(0x12, 0x80);
write_SCCB(0x11, 0x81);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x6a, 0x3e);
write_SCCB(0x3b, 0x09);
write_SCCB(0x13, 0xe0);
write_SCCB(0x01, 0x80);
write_SCCB(0x02, 0x80);
write_SCCB(0x00, 0x00);
write_SCCB(0x10, 0x00);
write_SCCB(0x13, 0xe5);
//
```

```
write_SCCB(0x39, 0x43); //50 for 30fps
write_SCCB(0x38, 0x12); //92 for 30fps
write_SCCB(0x37, 0x00);
write_SCCB(0x35, 0x91); //81 for 30fps
write_SCCB(0x0e, 0x20);
write_SCCB(0x1e, 0x04);
//
write_SCCB(0xa8, 0x80);
write_SCCB(0x12, 0x40);
write_SCCB(0x04, 0x00);
write_SCCB(0x0c, 0x04);
write_SCCB(0x0d, 0x80);
write_SCCB(0x18, 0xc6);
write_SCCB(0x17, 0x26);
write_SCCB(0x32, 0xad);
write_SCCB(0x03, 0x00);
write_SCCB(0x1a, 0x3d);
write_SCCB(0x19, 0x01);
write_SCCB(0x3f, 0xa6);
write_SCCB(0x14, 0x2e);
write_SCCB(0x15, 0x02);
write_SCCB(0x41, 0x02);
write_SCCB(0x42, 0x08);
//
write_SCCB(0x1b, 0x00);
write_SCCB(0x16, 0x06);
write_SCCB(0x33, 0xe2); //c0 for internal regulator
write_SCCB(0x34, 0xbf);
write_SCCB(0x96, 0x04);
write_SCCB(0x3a, 0x00);
write_SCCB(0x8e, 0x00);
//
write_SCCB(0x3c, 0x77);
write_SCCB(0x8b, 0x06);
write_SCCB(0x94, 0x88);
write_SCCB(0x95, 0x88);
write_SCCB(0x40, 0xc1);
write_SCCB(0x29, 0x3f); //2f for internal regulator
write_SCCB(0x0f, 0x42);
//
write_SCCB(0x3d, 0x92);
write_SCCB(0x69, 0x40);
write_SCCB(0x5c, 0xb9);
write_SCCB(0x5d, 0x96);
write_SCCB(0x5e, 0x10);
write_SCCB(0x59, 0xc0);
write_SCCB(0x5a, 0xaf);
write_SCCB(0x5b, 0x55);
write_SCCB(0x43, 0xf0);
write_SCCB(0x44, 0x10);
write_SCCB(0x45, 0x68);
write_SCCB(0x46, 0x96);
write_SCCB(0x47, 0x60);
write_SCCB(0x48, 0x80);
write_SCCB(0x5f, 0xe0);
write_SCCB(0x60, 0x8c); //0c for advanced AWB (related to lens)
write_SCCB(0x61, 0x20);
write_SCCB(0xa5, 0xd9);
```

```
write_SCCB(0xa4, 0x74);
write_SCCB(0x8d, 0x02);
write_SCCB(0x13, 0xe7);
//
write_SCCB(0x4f, 0x3a);
write_SCCB(0x50, 0x3d);
write_SCCB(0x51, 0x03);
write_SCCB(0x52, 0x12);
write_SCCB(0x53, 0x26);
write_SCCB(0x54, 0x38);
write_SCCB(0x55, 0x40);
write_SCCB(0x56, 0x40);
write_SCCB(0x57, 0x40);
write_SCCB(0x58, 0x0d);
//
write_SCCB(0x8c, 0x23);
write_SCCB(0x3e, 0x02);
write_SCCB(0xa9, 0xb8);
write_SCCB(0xaa, 0x92);
write_SCCB(0xab, 0x0a);
//
write_SCCB(0x8f, 0xdf);
write_SCCB(0x90, 0x00);
write_SCCB(0x91, 0x00);
write_SCCB(0x9f, 0x00);
write_SCCB(0xa0, 0x00);
write_SCCB(0x3a, 0x01);
//
write_SCCB(0x24, 0x70);
write_SCCB(0x25, 0x64);
write_SCCB(0x26, 0xc3);
//
write_SCCB(0x2a, 0x00); //10 for 50Hz
write_SCCB(0x2b, 0x00); //40 for 50Hz
//
//gamma
write_SCCB(0x6c, 0x40);
write_SCCB(0x6d, 0x30);
write_SCCB(0x6e, 0x4b);
write_SCCB(0x6f, 0x60);
write_SCCB(0x70, 0x70);
write_SCCB(0x71, 0x70);
write_SCCB(0x72, 0x70);
write_SCCB(0x73, 0x70);
write_SCCB(0x74, 0x60);
write_SCCB(0x75, 0x60);
write_SCCB(0x76, 0x50);
write_SCCB(0x77, 0x48);
write_SCCB(0x78, 0x3a);
write_SCCB(0x79, 0x2e);
write_SCCB(0x7a, 0x28);
write_SCCB(0x7b, 0x22);
write_SCCB(0x7c, 0x04);
write_SCCB(0x7d, 0x07);
write_SCCB(0x7e, 0x10);
write_SCCB(0x7f, 0x28);
write_SCCB(0x80, 0x36);
write_SCCB(0x81, 0x44);
```

```
write_SCCB(0x82, 0x52);
write_SCCB(0x83, 0x60);
write_SCCB(0x84, 0x6c);
write_SCCB(0x85, 0x78);
write_SCCB(0x86, 0x8c);
write_SCCB(0x87, 0x9e);
write_SCCB(0x88, 0xbb);
write_SCCB(0x89, 0xd2);
write_SCCB(0x8a, 0xe6);
```

13.3.2 SXGA mode

```
write_SCCB(0x12, 0x80);
write_SCCB(0x11, 0x80);
write_SCCB(0x6b, 0x0a);
write_SCCB(0x6a, 0x41);
write_SCCB(0x3b, 0x09);
write_SCCB(0x13, 0xe0);
write_SCCB(0x01, 0x80);
write_SCCB(0x02, 0x80);
write_SCCB(0x00, 0x00);
write_SCCB(0x10, 0x00);
write_SCCB(0x13, 0xe5);
//
write_SCCB(0x39, 0x43); //50 for 15fps
write_SCCB(0x38, 0x12); //93 for 15fps
write_SCCB(0x37, 0x00);
write_SCCB(0x35, 0x91); //81 for 15fps
write_SCCB(0x0e, 0x20);
write_SCCB(0x1e, 0x04);
//
write_SCCB(0xa8, 0x80);
write_SCCB(0x12, 0x00);
write_SCCB(0x04, 0x00);
write_SCCB(0x0c, 0x00);
write_SCCB(0x0d, 0x00);
write_SCCB(0x18, 0xbd);
write_SCCB(0x17, 0x1d);
write_SCCB(0x32, 0xad);
write_SCCB(0x03, 0x12);
write_SCCB(0x1a, 0x81);
write_SCCB(0x19, 0x01);
write_SCCB(0x14, 0x2e);
write_SCCB(0x15, 0x00);
write_SCCB(0x3f, 0xa6);
write_SCCB(0x41, 0x02);
write_SCCB(0x42, 0x08);
//
write_SCCB(0x1b, 0x00);
write_SCCB(0x16, 0x06);
write_SCCB(0x33, 0xe2); //c0 for internal regulator
write_SCCB(0x34, 0xbf);
write_SCCB(0x96, 0x04);
write_SCCB(0x3a, 0x00);
write_SCCB(0x8e, 0x00);
//
write_SCCB(0x3c, 0x77);
```

```
write_SCCB(0x8b, 0x06);
write_SCCB(0x94, 0x88);
write_SCCB(0x95, 0x88);
write_SCCB(0x40, 0xc1);
write_SCCB(0x29, 0x3f); //2f for internal regulator
write_SCCB(0x0f, 0x42);
//
write_SCCB(0x3d, 0x92);
write_SCCB(0x69, 0x40);
write_SCCB(0x5c, 0xb9);
write_SCCB(0x5d, 0x96);
write_SCCB(0x5e, 0x10);
write_SCCB(0x59, 0xc0);
write_SCCB(0x5a, 0xaf);
write_SCCB(0x5b, 0x55);
write_SCCB(0x43, 0xf0);
write_SCCB(0x44, 0x10);
write_SCCB(0x45, 0x68);
write_SCCB(0x46, 0x96);
write_SCCB(0x47, 0x60);
write_SCCB(0x48, 0x80);
write_SCCB(0x5f, 0xe0);
write_SCCB(0x60, 0x8c); //0c for advanced AWB (Related to lens)
write_SCCB(0x61, 0x20);
write_SCCB(0xa5, 0xd9);
write_SCCB(0xa4, 0x74);
write_SCCB(0x8d, 0x02);
write_SCCB(0x13, 0xe7);
//
write_SCCB(0x4f, 0x3a);
write_SCCB(0x50, 0x3d);
write_SCCB(0x51, 0x03);
write_SCCB(0x52, 0x12);
write_SCCB(0x53, 0x26);
write_SCCB(0x54, 0x38);
write_SCCB(0x55, 0x40);
write_SCCB(0x56, 0x40);
write_SCCB(0x57, 0x40);
write_SCCB(0x58, 0x0d);
//
write_SCCB(0x8c, 0x23);
write_SCCB(0x3e, 0x02);
write_SCCB(0xa9, 0xb8);
write_SCCB(0xaa, 0x92);
write_SCCB(0xab, 0x0a);
//
write_SCCB(0x8f, 0xdf);
write_SCCB(0x90, 0x00);
write_SCCB(0x91, 0x00);
write_SCCB(0x9f, 0x00);
write_SCCB(0xa0, 0x00);
write_SCCB(0x3a, 0x01);
//
write_SCCB(0x24, 0x70);
write_SCCB(0x25, 0x64);
write_SCCB(0x26, 0xc3);
//
write_SCCB(0x2a, 0x00); //10 for 50Hz
```

```
write_SCCB(0x2b, 0x00); //34 for 50Hz
//
//gamma
write_SCCB(0x6c, 0x40);
write_SCCB(0x6d, 0x30);
write_SCCB(0x6e, 0x4b);
write_SCCB(0x6f, 0x60);
write_SCCB(0x70, 0x70);
write_SCCB(0x71, 0x70);
write_SCCB(0x72, 0x70);
write_SCCB(0x73, 0x70);
write_SCCB(0x74, 0x60);
write_SCCB(0x75, 0x60);
write_SCCB(0x76, 0x50);
write_SCCB(0x77, 0x48);
write_SCCB(0x78, 0x3a);
write_SCCB(0x79, 0x2e);
write_SCCB(0x7a, 0x28);
write_SCCB(0x7b, 0x22);
write_SCCB(0x7c, 0x04);
write_SCCB(0x7d, 0x07);
write_SCCB(0x7e, 0x10);
write_SCCB(0x7f, 0x28);
write_SCCB(0x80, 0x36);
write_SCCB(0x81, 0x44);
write_SCCB(0x82, 0x52);
write_SCCB(0x83, 0x60);
write_SCCB(0x84, 0x6c);
write_SCCB(0x85, 0x78);
write_SCCB(0x86, 0x8c);
write_SCCB(0x87, 0x9e);
write_SCCB(0x88, 0xbb);
write_SCCB(0x89, 0xd2);
write_SCCB(0x8a, 0xe6);
```

13.3.3 VGA change to SXGA

```
write_SCCB(0x11, 0x80);
write_SCCB(0x12, 0x00);
write_SCCB(0x0c, 0x00);
write_SCCB(0x0d, 0x00);
write_SCCB(0x18, 0xbe);
write_SCCB(0x17, 0x1e);
write_SCCB(0x32, 0xbf);
write_SCCB(0x03, 0x12);
write_SCCB(0x1a, 0x81);
write_SCCB(0x19, 0x01);

write_SCCB(0x2a, 0x10);//24M clock
write_SCCB(0x2b, 0x34);//24M clock
write_SCCB(0x6a, 0x41);//24M clock
```

13.3.4 SXGA change to VGA

```
write_SCCB(0x11, 0x81);
write_SCCB(0x12, 0x40);
write_SCCB(0x0c, 0x04);
write_SCCB(0x0d, 0x80);
write_SCCB(0x18, 0xc7);
```

```
write_SCCB(0x17, 0x27);
write_SCCB(0x32, 0xad);
write_SCCB(0x03, 0x00);
write_SCCB(0x1a, 0x3d);
write_SCCB(0x19, 0x01);
write_SCCB(0x6a, 0x3e);

write_SCCB(0x11, 0x81);//for 24M clock
write_SCCB(0x2a, 0x10);//for 24M clock
write_SCCB(0x2b, 0x40);//for 24M clock
```

14. Capture Sequence

14.1 Stop AEC

write_SCCB(0x13, 0xe2);

14.2 Read register Value

Read Gain

Gain = Reg0x00

Read Exposure: 4 Bytes

Preview_Exposure = (Reg0xa1 & 0x3f)<<10 + Reg0x10<<2 + Reg0x04 & 0x3

Read Reg06= Reg0x06

14.3 Calculate Capture Exposure from preview

$$Capture\_Exposure = Preview\_Exposure * \frac{Capture\_Framerate * Capture\_MaxLines}{Preview\_Framerate * Preview\_MaxLines}$$

14.4 Redistribute Exposure/Gain with target brightness unchanged

Capture_Gain = (Gain[7]+1)*(Gain[6]+1)*(Gain[5]+1)*(Gain[4]+1)*(Gain[3:0] + 16)

```
Gain = Reg0x00
Capture_gain = Gain & 0x0f + 16
If (Gain & 0x10)
        Capture_Gain = Capture_Gain << 1;
If (Gain & 0x20)
        Capture_Gain = Capture_Gain << 1;
If (Gain & 0x40)
        Capture_Gain = Capture_Gain << 1;
If (Gain & 0x80)
        Capture_Gain = Capture_Gain << 1;
```

Capture_Exposure_Gain = Capture_Exposure * Capture_Gain* Target_Luminance / Reg0x06

if(reg0x10==0x7c)

```
{
        if(Reg0x06 >reg0x25)
        {
                Capture_Exposure_Gain = Capture_Exposure_Gain *84/100;
        }
        else
        {
                Capture_Exposure_Gain = Capture_Exposure_Gain *90/100;
        }
}
//Lines_10ms = number of lines equal to 1/100 seconds for 50hz, 1/120 seconds for 60hz

        if (Capture_Exposure_Gain < Max_Exposure_lines*16)
        {
                Capture_Exposure = Capture_Exposure_Gain/16;
                If (Capture_Exposure > Lines_10ms) {
                        //Capture_Exposure = n*Lines_10ms, Banding removed
                        Capture_Expsoure /= Lines_10ms;
                        Capture_Expsoure *= Lines_10ms;
                }

        }
        else
        {
                Capture_Exposure = Max_Exposure_Lines;
        }
    if (capture_exposure == 0) { capture_exposure = 1; }
        Capture_Gain = (Capture_Exposure_Gain*2/Capture_Exposure + 1)/2;
        if(reg0x00>0x1f)
  {
          reg0x00=0x1f
  }
        If (Capture_Gain < 31) {
                60 8c 20; Turn off white Pixel correction
        }
```

14.5 write back the gain/exposure value

Reg0x04 = Reg0x04 & 0xfc | capture_expsoure & 0x03
Reg0x10 = (Capture_Exposure >> 2) & 0xff
Reg0xa1 = Capture_Expsure >> 10

Reg0x00 = Calculate back gain to register value (Capture_Gain = (0x00[7]+1)*(0x00[6]+1)*(0x00[5]+1)*(0x00[4]+1)*(0x00[3:0]/16 + 1))

```
        Gain = 0
        If (Capture_Gain > 31)
```

```
{
        Gain |= 0x10;
        Capture_Gain = Capture_Gain >> 1;
}
If (Capture_Gain > 31)
{
        Gain |= 0x20;
        Capture_Gain = Capture_Gain >> 1;
}
If (Capture_Gain > 31)
{
        Gain |= 0x40;
        Capture_Gain = Capture_Gain >> 1;
}
If (Capture_Gain > 31)
{
        Gain |= 0x80;
        Capture_Gain = Capture_Gain >> 1;
}

if (Capture_Gain > 16) {
        Gain |= ((Capture_Gain -16) & 0x0f)
}
```

14.6 Set to SXGA

Write SXGA tegister

14.7 Wait for 2 Vsyncs, capture the third frame

14.8 Enable AEC

write_SCCB(0x13, 0xe7);