

## OVM7690 1/13” VGA CameraCube Camera Module Software Application Note

Last Modified: Nov 13<sup>th</sup>, 2008

Document Revision: 1.00

OmniVision Technologies, Inc. reserves the right to make changes without further notice to any product herein to improve reliability, function or design. OmniVision does not assume any liability arising out of the application or use of any project, circuit described herein; neither does it convey any license under its patent nor the right of others.

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

---

This document contains information of a proprietary nature. None of this information shall be divulged to persons other than OmniVision Technologies, Inc. employee authorized by the nature of their duties to receive such information, or individuals or organizations authorized by OmniVision Technologies, Inc.



## Table of Contents

1. Select Output format.....	4
1.1 Backend with full ISP.....	4
1.2 Backend with YCbCr ISP.....	4
1.3 Backend without ISP.....	4
1.4 Equations to Convert from One Format to Another.....	5
2. Select Output Resolution?.....	6
2.1 Backend with ISP.....	6
2.2 Backend without ISP.....	6
3. Adjust frame rate.....	6
3.1 Frame Rate Adjustment for 24Mhz input clock.....	6
30 fps, PCLK = 24Mhz.....	6
15 fps, PCLK = 12Mhz.....	6
25fps, PCLK = 24Mhz.....	7
12.5fps, PCLK = 12Mhz.....	7
3.2 Frame Rate Adjustment for 26 Mhz input clock.....	7
30 fps, PCLK = 26Mhz.....	7
15 fps, PCLK = 13Mhz.....	7
25fps, PCLK = 26Mhz.....	8
12.5fps, PCLK = 13Mhz.....	8
3.3 Frame rate adjustment for 13 Mhz input clock.....	8
30 fps, PCLK = 26Mhz.....	8
15 fps, PCLK = 13Mhz.....	9
25fps, PCLK = 26Mhz.....	9
12.5fps, PCLK = 13Mhz.....	9
4. Night Mode.....	9
4.1 Night Mode with Fixed Frame Rate.....	10
For 24Mhz/26Mhz Clock Input.....	10
For 13Mhz Clock Input.....	10
4.2 Night Mode with Auto Frame Rate.....	10
For 24Mhz/26Mhz Clock Input.....	10
For 13Mhz Clock Input.....	11
5. Remove Light Band.....	12
5.1 Light Band.....	12
5.2 Remove Light band.....	12
5.3 Select Banding Filter by Region Information.....	12
Banding Filter Setting for 24Mhz Input Clock .....	13
Banding Filter Setting for 13Mhz/26Mhz Input Clock .....	13
5.4 Select Banding Filter by Automatic Light Frequency Detection.....	14
Banding Filter Setting for 24Mhz Input Clock.....	14
Banding Filter Setting for 13Mhz/26Mhz Input Clock.....	15
5.5 When Light Band can not be Removed.....	15
6. White Balance.....	15
6.1 Simple White Balance.....	15



6.2 Advanced White Balance.....	16
7. Defect Pixel Correction.....	16
8. BLC.....	17
9. Video Mode.....	17
10. Digital zoom.....	17
11. OV7690 Functions.....	17
11.1 Light Mode.....	17
11.2 Color Saturation.....	18
11.3 Exposure Value.....	20
11.4 Contrast.....	21
11.5 Special effects.....	24
12. Deal with Lens.....	25
12.1 Light fall off.....	25
12.2 Dark corner.....	25
12.3 Resolution.....	26
12.4 Optical contrast.....	26
12.5 Lens Cover.....	26
12.6 Lens Correction.....	26
13. Reference Settings.....	26
13.1 YCbCr.....	26
13.2 RGB raw.....	29
13.3 RGB565.....	29
13.4 Size.....	29
Revision History.....	32

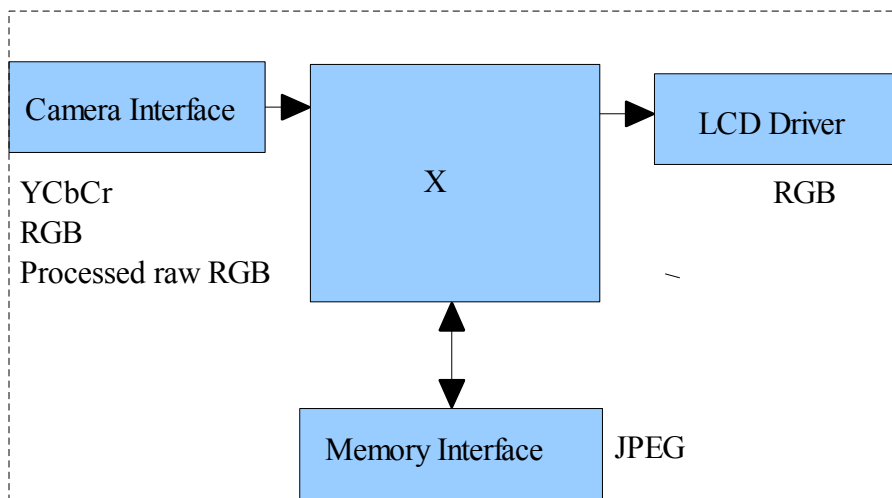
## 1. Select Output format

OVM7690 support following output formats:

YCbCr, RGB565, CCIR656 and raw RGB.

How to choose the right output format for camera phone design or other applications? Let's look at the backend chip first.

The general diagram of backend chip is as below:



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

### 1.1 Backend with full ISP

This kind of backend has full ISP. It takes raw RGB input, doing interpolation to generate RGB24 and doing translation to generate YCbCr.

### 1.2 Backend with YCbCr ISP

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

### 1.3 Backend without ISP

This kind of backend doesn't have ISP built-in. It can not convert from one format to another by hardware. Actually the format conversion is done by software. There are 3 possible solution for this kind of backend chips.

- Sensor output YCbCr. Backend convert YCbCr to RGB for display by software.
- Sensor output RGB565. Backend convert RGB565 to YCbCr for compression.
- 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.

Solution c. provide similar picture quality as solution a. But since preview is RGB565, capture is YCbCr, the captured picture may looks a little different than preview image.

### 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. Select Output Resolution?

### 2.1 Backend with ISP

If Backend chip has built-in ISP (Full ISP or YCbCr ISP), the ISP could do image scale. So OV7690 outputs only VGA format. ISP scaled VGA image to other resolution that mobile device needed.

### 2.2 Backend without ISP

If backend chip doesn't have image scale capability, then the LCD scaler of OV7690 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.

## 3. Adjust frame rate

The recommended frame rates for preview and capture are 30fps and 15fps for 60Hz light environment, 25fps and 14.3fps for 50Hz light environment. The recommended frame rate for video is always 15fps, not matter light environment is 50Hz or 60Hz. 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 Frame Rate Adjustment for 24Mhz input clock

#### 30 fps, PCLK = 24Mhz

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x00);  
write_SCCB(0x29, 0x50);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);  
write_SCCB(0x2c, 0x00);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

#### 15 fps, PCLK = 12Mhz

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x01);  
write_SCCB(0x29, 0x50);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);
```

```
write_SCCB(0x2c, 0x00);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

### **25fps, PCLK = 24Mhz**

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x00);  
write_SCCB(0x29, 0x50);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);  
write_SCCB(0x2c, 0x67);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

### **12.5fps, PCLK = 12Mhz**

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x01);  
write_SCCB(0x29, 0x50);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);  
write_SCCB(0x2c, 0x67);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

## **3.2 Frame Rate Adjustment for 26 Mhz input clock**

### **30 fps, PCLK = 26Mhz**

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x00);  
write_SCCB(0x29, 0x50);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);  
write_SCCB(0x2c, 0x2b);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

**15 fps, PCLK = 13Mhz**

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x29, 0x50);
write_SCCB(0x2a, 0x30);
write_SCCB(0x2b, 0x08);
write_SCCB(0x2c, 0x2b);
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);
```

**25fps, PCLK = 26Mhz**

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x29, 0x50);
write_SCCB(0x2a, 0x30);
write_SCCB(0x2b, 0x08);
write_SCCB(0x2c, 0x9a);
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);
```

**12.5fps, PCLK = 13Mhz**

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x29, 0x50);
write_SCCB(0x2a, 0x30);
write_SCCB(0x2b, 0x08);
write_SCCB(0x2c, 0x9a);
write_SCCB(0x15, 0x00);
write_SCCB(0x2d, 0x00);
write_SCCB(0x2e, 0x00);
```

**3.3 Frame rate adjustment for 13 Mhz input clock**

**30 fps, PCLK = 26Mhz**

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x29, 0x10);
write_SCCB(0x2a, 0x30);
write_SCCB(0x2b, 0x08);
write_SCCB(0x2c, 0x2b);
```

```
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

### **15 fps, PCLK = 13Mhz**

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x00);  
write_SCCB(0x29, 0x50);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);  
write_SCCB(0x2c, 0x2b);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

### **25fps, PCLK = 26Mhz**

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x00);  
write_SCCB(0x29, 0x10);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);  
write_SCCB(0x2c, 0x9a);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

### **12.5fps, PCLK = 13Mhz**

```
SCCB_salve_Address = 0x42;  
write_SCCB(0x11, 0x00);  
write_SCCB(0x29, 0x50);  
write_SCCB(0x2a, 0x30);  
write_SCCB(0x2b, 0x08);  
write_SCCB(0x2c, 0x9a);  
write_SCCB(0x15, 0x00);  
write_SCCB(0x2d, 0x00);  
write_SCCB(0x2e, 0x00);
```

## **4. Night Mode**

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

rate is decreased to 3.75fps.

## 4.1 Night Mode with Fixed Frame Rate

### ***For 24Mhz/26Mhz Clock Input***

3.75fps night mode for 60Hz light environment

```
SCCB_salve_Address = 0x42;
```

```
write_SCCB(0x11, 0x07);
```

```
write_SCCB(0x29, 0x50);
```

```
write_SCCB(0x15, 0x00);
```

```
write_SCCB(0x2d, 0x00);
```

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

3.125fps night mode for 50Hz light environment

```
SCCB_salve_Address = 0x42;
```

```
write_SCCB(0x11, 0x07);
```

```
write_SCCB(0x29, 0x50);
```

```
write_SCCB(0x15, 0x00);
```

```
write_SCCB(0x2d, 0x00);
```

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

### ***For 13Mhz Clock Input***

3.75fps night mode for 60Hz light environment

```
SCCB_salve_Address = 0x42;
```

```
write_SCCB(0x11, 0x03);
```

```
write_SCCB(0x29, 0x50);
```

```
write_SCCB(0x15, 0x00);
```

```
write_SCCB(0x2d, 0x00);
```

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

3.125fps night mode for 50Hz light environment

```
SCCB_salve_Address = 0x42;
```

```
write_SCCB(0x11, 0x03);
```

```
write_SCCB(0x29, 0x50);
```

```
write_SCCB(0x15, 0x00);
```

```
write_SCCB(0x2d, 0x00);
```

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

## 4.2 Night Mode with Auto Frame Rate

### ***For 24Mhz/26Mhz Clock Input***

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

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x15, 0xcc);
```

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

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x15, 0xb8);
```

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

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x15, 0xcc);
```

12.5fps ~ 3.125fps night mode for 50Hz light environment

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x01);
write_SCCB(0x15, 0xb8);
```

### **For 13Mhz Clock Input**

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

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x29, 0x70);
write_SCCB(0x15, 0xcc);
```

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

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x29, 0x50);
write_SCCB(0x15, 0xb8);
```

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

```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x29, 0x70);
write_SCCB(0x15, 0xcc);
```

12.5fps ~ 3.125fps night mode for 50Hz light environment

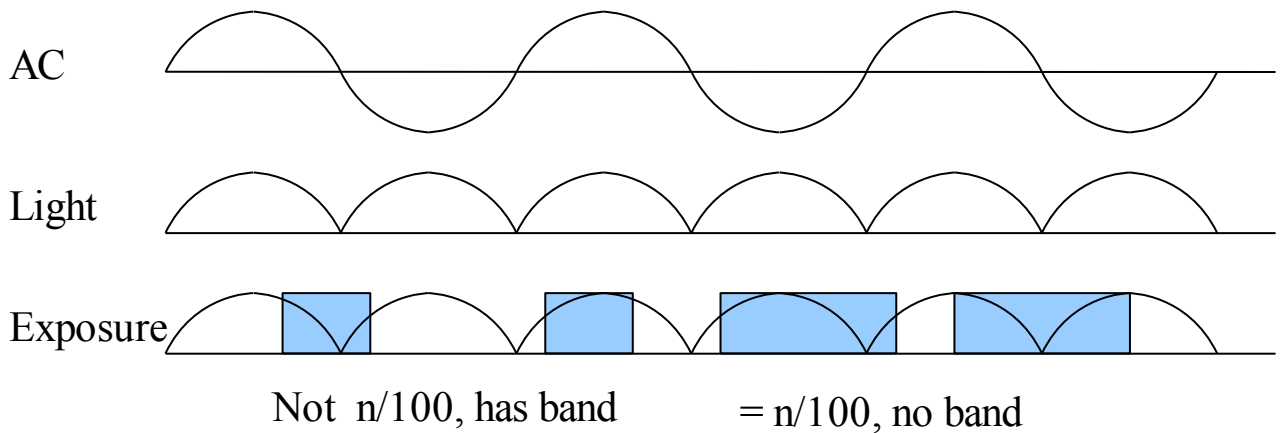
```
SCCB_salve_Address = 0x42;
write_SCCB(0x11, 0x00);
write_SCCB(0x29, 0x50);
write_SCCB(0x15, 0xb8);
```

Note:

When OVM7690 is set to low frame rate, there may be many white pixels shown on LCD of mobile phone or on PC display.

## 5. Remove Light Band

### 5.1 Light Band



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



### 5.2 Remove Light band

Light band is removed by set exposure to  $n/100$  ( $n/120$  for 60Hz)seconds. The banding filter value tell OV7690 how many lines is  $1/100$  ( $1/120$  for 60Hz) seconds.

### 5.3 Select Banding Filter by Region Information

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

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

#### ***Banding Filter Setting for 24Mhz Input Clock***

```
30fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x9a); //50Hz banding filter
write_SCCB(0x51, 0x80); //60Hz banding filter
write_SCCB(0x21, 0x34); //3 step for 50hz, 4 step for 60hz
write_SCCB(0x14, 0x20); //Select 60Hz banding filter
```

```
15fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x4c); //50Hz banding filter
write_SCCB(0x51, 0x3f); //60Hz banding filter
write_SCCB(0x21, 0x68); //6 step for 50hz, 8 step for 60hz
write_SCCB(0x14, 0x20); //Select 60Hz banding filter
```

```
25fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x9a); //50Hz banding filter
write_SCCB(0x51, 0x80); //60Hz banding filter
write_SCCB(0x21, 0x34); //4 step for 50hz, 4 step for 60hz
write_SCCB(0x14, 0x21); //Select 50Hz banding filter
```

```
12.5fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x4c); //50Hz banding filter
write_SCCB(0x51, 0x3f); //60Hz banding filter
write_SCCB(0x21, 0x68); //6 step for 50hz, 8 step for 60hz
write_SCCB(0x14, 0x21); //Select 50Hz banding filter
```

#### ***Banding Filter Setting for 13Mhz/26Mhz Input Clock***

```
30fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
```

```
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0xa6); //50Hz banding filter
write_SCCB(0x51, 0x8a); //60Hz banding filter
write_SCCB(0x21, 0x34); //3 step for 50hz, 4 step for 60hz
write_SCCB(0x14, 0x20); //Select 60Hz banding filter
```

```
15fps for 60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x53); //50Hz banding filter
write_SCCB(0x51, 0x45); //60Hz banding filter
write_SCCB(0x21, 0x68); //6 step for 50hz, 8 step for 60hz
write_SCCB(0x14, 0x20); //Select 60Hz banding filter
```

```
25fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0xa6); //50Hz banding filter
write_SCCB(0x51, 0x8a); //60Hz banding filter
write_SCCB(0x21, 0x34); //3 step for 50hz, 4 step for 60hz
write_SCCB(0x14, 0x21); //Select 50Hz banding filter
```

```
12.5fps for 50Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x53); //50Hz banding filter
write_SCCB(0x51, 0x45); //60Hz banding filter
write_SCCB(0x21, 0x68); //6 step for 50hz, 8 step for 60hz
write_SCCB(0x14, 0x21); //Select 50Hz banding filter
```

## 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. OVM7690 could automatic select 50Hz or 60Hz banding filter based on light frequency detection. Please contact with OmniVision local FAE for automatic light frequency detection settings.

### ***Banding Filter Setting for 24Mhz Input Clock***

```
30fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x99); //50Hz banding filter
write_SCCB(0x51, 0x7f); //60Hz banding filter
write_SCCB(0x21, 0x34); //3 step for 50hz, 4 step for 60hz
write_SCCB(0x14, 0xb2); //Auto detect banding filter
```

```
15fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x4c); //50Hz banding filter
write_SCCB(0x51, 0x3f); //60Hz banding filter
write_SCCB(0x21, 0x68); //6 step for 50hz, 8 step for 60hz
write_SCCB(0x14, 0xb2); //Auto detect banding filter
```

### ***Banding Filter Setting for 13Mhz/26Mhz Input Clock***

```
30fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0xa6); //50Hz banding filter
write_SCCB(0x51, 0x8a); //60Hz banding filter
write_SCCB(0x21, 0x34); //3 step for 50hz, 4 step for 60hz
write_SCCB(0x14, 0xb2); // auto detect banding filter
```

```
15fps for 50/60Hz light frequency
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xef); //banding filter enable
write_SCCB(0x50, 0x53); //50Hz banding filter
write_SCCB(0x51, 0x45); //60Hz banding filter
write_SCCB(0x21, 0x68); //6 step for 50hz, 8 step for 60hz
write_SCCB(0x14, 0xb2); // auto detect banding filter
```

## **5.5 When Light Band can not be Removed**

Normally the light band is removed by banding filter.

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

The light band is this conditions could not be removed for all CMOS sensors, not only OV7690. So there is no way to remove light band in this condition.

## **6. White Balance**

OVM7690 support simple white balance and advanced white 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 = 0x42;  
write_SCCB(0x13, 0xef); //AWB on  
write_SCCB(0x8e, 0x92); //enable 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.

**Note: For the customers to get wonderful image quality, OmniVision suggest to use advanced white balance.**

## 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 than normal pixels, but not complete white. Black wounded pixels are much darker than normal pixels, but not complete black.

OVM7690 has built-in defect pixel correction function. If OVM7690 output YCbCr, RGB565, the defect pixel correction function could be enabled to fix defect pixels. But if Bayer raw RGB is used, the defect pixel correction function of sensor could not be used. The defect pixel correction of backend chip should be used instead.

Please pay attention to the defect pixel correction function of backend chip. Some backend chip may not be able to correct all defect pixels of OVM7690.

## **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 OVM7690. 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 OV7690 output image smaller than QVGA, it may support digital zoom. For example

VGA	not digital zoom supported
QVGA	1x, 2x
QQVGA	1x, 2x, 4x
QCIF	1x, 1.8x
QQCIF	1x, 2x, 3.6x

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

## **11. OV7690 Functions**

### **11.1 Light Mode**

Auto

SCCB\_salve\_Address = 0x42;

```
write_SCCB(0x13, 0xf7); //AWB on
write_SCCB(0x15, 0x00);
```

#### Sunny

```
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xf5); //AWB off
write_SCCB(0x01, 0x5a);
write_SCCB(0x02, 0x5c);
write_SCCB(0x15, 0x00);
```

#### Cloudy

```
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xf5); //AWB off
write_SCCB(0x01, 0x58);
write_SCCB(0x02, 0x60);
write_SCCB(0x15, 0x00);
```

#### Office

```
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xf5); //AWB off
write_SCCB(0x01, 0x84);
write_SCCB(0x02, 0x4c);
write_SCCB(0x15, 0x00);
```

#### Home

```
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xf5); //AWB off
write_SCCB(0x01, 0x96);
write_SCCB(0x02, 0x40);
write_SCCB(0x15, 0x00);
```

#### Night

```
SCCB_salve_Address = 0x42;
write_SCCB(0x13, 0xf7); //AWB on
write_SCCB(0x15, 0xb8);
```

## 11.2 Color Saturation

The color saturation of OV7690 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. Set

0xd2.bit1='1' to enable.

Saturation + 4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x80);
write_SCCB(0xD9, 0x80);
write_SCCB(0xD2, 0x02);
```

Saturation + 3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x70);
write_SCCB(0xD9, 0x70);
write_SCCB(0xD2, 0x02);
```

Saturation + 2

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x60);
write_SCCB(0xD9, 0x60);
write_SCCB(0xD2, 0x02);
```

Saturation + 1

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x50);
write_SCCB(0xD9, 0x50);
write_SCCB(0xD2, 0x02);
```

Saturation 0

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x40);
write_SCCB(0xD9, 0x40);
write_SCCB(0xD2, 0x00);
```

#### Saturation -1

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x30);
write_SCCB(0xD9, 0x30);
write_SCCB(0xD2, 0x02);
```

#### Saturation - 2

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x20);
write_SCCB(0xD9, 0x20);
write_SCCB(0xD2, 0x02);
```

#### Saturation - 3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x10);
write_SCCB(0xD9, 0x10);
write_SCCB(0xD2, 0x02);
```

#### Saturation - 4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xD8, 0x00);
write_SCCB(0xD9, 0x00);
write_SCCB(0xD2, 0x02);
```

### 11.3 Exposure Value

The EV of OV7690 could be adjusted. Higher EV will make the picture more bright.

#### EV +4

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24, 0xb8);
write_SCCB(0x25, 0xb0);
write_SCCB(0x26, 0xf8);
```

EV +3

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0xa8);
write_SCCB(0x25,0xa0);
write_SCCB(0x26,0xe5);
```

EV +2

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0x98);
write_SCCB(0x25,0x90);
write_SCCB(0x26,0xd6);
```

EV +1

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0x88);
write_SCCB(0x25,0x80);
write_SCCB(0x26,0xc5);
```

EV 0

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0x78);
write_SCCB(0x25,0x68);
write_SCCB(0x26,0xB4);
```

EV -1

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0x60);
write_SCCB(0x25,0x58);
write_SCCB(0x26,0x92);
```

EV -2

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0x50);
write_SCCB(0x25,0x48);
write_SCCB(0x26,0x92);
```

EV -3

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0x40);
write_SCCB(0x25,0x38);
write_SCCB(0x26,0x71);
```

EV -4

```
SCCB_salve_Address = 0x42;
write_SCCB(0x24,0x30);
write_SCCB(0x25,0x28);
write_SCCB(0x26,0x61);
```

## 11.4 Contrast

The contrast of OV7689 could be adjusted. Higher contrast will make the picture sharp. But the side effect is losing dynamic range.

Contrast +4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x30);
write_SCCB(0xd3, 0x00);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp &= 0xfb;
write_SCCB(0xdc, temp);
```

Contrast +3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x2c);
write_SCCB(0xd3, 0x00);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp &= 0xfb;
write_SCCB(0xdc, temp);
```

Contrast +2

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x28);
write_SCCB(0xd3, 0x00);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp &= 0xfb;
write_SCCB(0xdc, temp);
```

Contrast +1

```

SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x24);
write_SCCB(0xd3, 0x00);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp &= 0xfb;
write_SCCB(0xdc, temp);
    
```

Contrast 0

```

SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x20);
write_SCCB(0xd3, 0x00);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp &= 0xfb;
write_SCCB(0xdc, temp);
    
```

Contrast -1

```

SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x1c);
write_SCCB(0xd3, 0x20);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp |= 0x04;
write_SCCB(0xdc, temp);
    
```

Contrast -2

```

SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x18);
write_SCCB(0xd3, 0x48);
    
```

```
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp |= 0x04;
write_SCCB(0xdc, temp);
```

#### Contrast -3

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x14);
write_SCCB(0xd3, 0x80);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp |= 0x04;
write_SCCB(0xdc, temp);
```

#### Contrast -4

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x33;
write_SCCB(0x81, temp);
write_SCCB(0xd5, 0x20);
write_SCCB(0xd4, 0x10);
write_SCCB(0xd3, 0xd0);
write_SCCB(0xd2, 0x04);
temp = read_SCCB(0xdc);
temp |= 0x04;
write_SCCB(0xdc, temp);
```

## 11.5 Special effects

OVM7690 support some special effects such as B/W, negative, sepia, bluish, redish, greenish etc. If users need other special effects, it should be supported by backend chips.

#### Normal

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp &= 0xdf;
write_SCCB(0x81, temp);
write_SCCB(0x28, 0x00);
write_SCCB(0xd2, 0x00);
```

#### Antique

```
SCCB_salve_Address = 0x42;
```

```
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
write_SCCB(0x28, 0x00);
write_SCCB(0xd2, 0x18);
write_SCCB(0xda, 0x40);
write_SCCB(0xdb, 0xa0);
```

#### Bluish

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
write_SCCB(0x28, 0x00);
write_SCCB(0xd2, 0x18);
write_SCCB(0xda, 0xa0);
write_SCCB(0xdb, 0x40);
```

#### Greenish

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
write_SCCB(0x28, 0x00);
write_SCCB(0xd2, 0x18);
write_SCCB(0xda, 0x60);
write_SCCB(0xdb, 0x60);
```

#### Redish

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
write_SCCB(0x28, 0x00);
write_SCCB(0xd2, 0x18);
write_SCCB(0xda, 0x80);
write_SCCB(0xdb, 0xc0);
```

#### B&W

```
SCCB_salve_Address = 0x42;
temp = read_SCCB(0x81);
temp |= 0x20;
write_SCCB(0x81, temp);
write_SCCB(0x28, 0x00);
write_SCCB(0xd2, 0x18);
write_SCCB(0xda, 0x80);
write_SCCB(0xdb, 0x80);
```

Negative

```
SCCB_salve_Address = 0x42;  
temp = read_SCCB(0x81);  
temp |= 0x20;  
write_SCCB(0x81, temp);  
write_SCCB(0x28, 0x80);  
write_SCCB(0xd2, 0x00);
```

## **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 OV7690 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 OVM7690 the focus distance is about 40cm. The depth of field is about from 20~200cm to infinite. If checking resolution of camera module, the resolution chart should be placed 40 cm away.

### **12.4 Optical contrast**

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

### **12.5 Lens Cover**

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

## 12.6 Lens Correction

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

## 13. Reference Settings

### 13.1 YCbCr

```
//OV7690
// MCLK = 24MHz
// 14.3fps, 50Hz
write_SCCB(0x12, 0x80);

delay(2ms);

write_SCCB(0x0c,0xd6);
write_SCCB(0x48,0x42);
write_SCCB(0x27,0x80);
write_SCCB(0x64,0x10);
write_SCCB(0x68,0xb4);
write_SCCB(0x69,0x12);
write_SCCB(0x2f,0x60);
write_SCCB(0x41,0x43);
write_SCCB(0x44,0x24);
write_SCCB(0x4b,0x0e);
write_SCCB(0x4c,0x7b);
write_SCCB(0x4d,0x0a);
write_SCCB(0x29,0x50);
write_SCCB(0x1b,0x19);
write_SCCB(0x39,0x80);
write_SCCB(0x80,0x7f);
write_SCCB(0x81,0xff);
write_SCCB(0x91,0x20);
write_SCCB(0x21,0x44);
write_SCCB(0x11,0x01);
write_SCCB(0x12,0x00);
write_SCCB(0x82,0x03);
write_SCCB(0xd0,0x248);
write_SCCB(0x2B,0x38);
write_SCCB(0x15,0x14);
write_SCCB(0x16,0x03);
write_SCCB(0x17,0x69);
write_SCCB(0x18,0xa4);
write_SCCB(0x19,0x0c);
write_SCCB(0x1a,0xf6);
```



```

write_SCCB(0x3e,0x30);
write_SCCB(0xc8,0x02);
write_SCCB(0xc9,0x80);
write_SCCB(0xca,0x01);
write_SCCB(0xcb,0xe0);
write_SCCB(0xcc,0x02);
write_SCCB(0xcd,0x80);
write_SCCB(0xce,0x01);
write_SCCB(0xcf,0xe0);
write_SCCB(0x80,0x7F);
write_SCCB(0x85,0x10);
write_SCCB(0x86,0x00);
write_SCCB(0x87,0x00);
write_SCCB(0x88,0x00);
write_SCCB(0x89,0x35);
write_SCCB(0x8a,0x30);
write_SCCB(0x8b,0x33);
write_SCCB(0xbb,0xbe);
write_SCCB(0xbc,0xc0);
write_SCCB(0xbd,0x02);
write_SCCB(0xbe,0x16);
write_SCCB(0xbf,0xc2);
write_SCCB(0xc0,0xd9);
write_SCCB(0xc1,0x1e);
write_SCCB(0xb4,0x36);
write_SCCB(0xb5,0x06);
write_SCCB(0xb7,0x00);
write_SCCB(0xb6,0x04);
write_SCCB(0xb8,0x06);
write_SCCB(0xb9,0x02);
write_SCCB(0xba,0x00);
write_SCCB(0x24,0x78);
write_SCCB(0x25,0x68);
write_SCCB(0x26,0xB4);
write_SCCB(0x81,0xff);
write_SCCB(0x5a,0x30);
write_SCCB(0x5b,0xa5);
write_SCCB(0x5c,0x30);
write_SCCB(0x5d,0x20);
write_SCCB(0xa3,0x05);
write_SCCB(0xa4,0x10);
write_SCCB(0xa5,0x25);
write_SCCB(0xa6,0x46);
write_SCCB(0xa7,0x57);
write_SCCB(0xa8,0x64);
write_SCCB(0xa9,0x70);
write_SCCB(0xaa,0x7c);

```

```

write_SCCB(0xab,0x87);
write_SCCB(0xac,0x90);
write_SCCB(0xad,0x9f);
write_SCCB(0xae,0xac);
write_SCCB(0xaf,0xc1);
write_SCCB(0xb0,0xd5);
write_SCCB(0xb1,0xe7);
write_SCCB(0xb2,0x21);
write_SCCB(0x8c,0x5c);
write_SCCB(0x8d,0x11);
write_SCCB(0x8e,0x12);
write_SCCB(0x8f,0x19);
write_SCCB(0x90,0x50);
write_SCCB(0x91,0x21);
write_SCCB(0x92,0x9c);
write_SCCB(0x93,0x9b);
write_SCCB(0x94,0x0c);
write_SCCB(0x95,0x0d);
write_SCCB(0x96,0xff);
write_SCCB(0x97,0x00);
write_SCCB(0x98,0x3f);
write_SCCB(0x99,0x30);
write_SCCB(0x9a,0x4d);
write_SCCB(0x9b,0x3d);
write_SCCB(0x9c,0xf0);
write_SCCB(0x9d,0xf0);
write_SCCB(0x9e,0xf0);
write_SCCB(0x9f,0xff);
write_SCCB(0xa0,0x5f);
write_SCCB(0xa1,0x61);
write_SCCB(0xa2,0x0c);
write_SCCB(0x14,0x21);
write_SCCB(0x13,0xf7);
    
```

## 13.2 RGB raw

Contact with OmniVision local FAE.

## 13.3 RGB565

Contact with OmniVision local FAE

### 13.4 Size

VGA 640\*480:

```
SCCB_salve_Address = 0x42;
write_SCCB(0x16,0x03);
write_SCCB(0x17,0x69);
write_SCCB(0x18,0xa4);
write_SCCB(0x19,0x0c);
write_SCCB(0x1a,0xf6);
write_SCCB(0x22,0x00);
write_SCCB(0xc8,0x02);
write_SCCB(0xc9,0x80);
write_SCCB(0xca,0x01);
write_SCCB(0xcb,0xe0);
write_SCCB(0xcc,0x02);
write_SCCB(0xcd,0x80);
write_SCCB(0xce,0x01);
write_SCCB(0xcf,0xe0);
```

QVGA 320\*240:

```
SCCB_salve_Address = 0x42;
write_SCCB(0x16,0x03);
write_SCCB(0x17,0x69);
write_SCCB(0x18,0xa4);
write_SCCB(0x19,0x06);
write_SCCB(0x1a,0xf6);
write_SCCB(0x22,0x10);
write_SCCB(0xc8,0x02);
write_SCCB(0xc9,0x80);
write_SCCB(0xca,0x00);
write_SCCB(0xcb,0xf0);
write_SCCB(0xcc,0x01);
write_SCCB(0xcd,0x40);
write_SCCB(0xce,0x00);
write_SCCB(0xcf,0xf0);
```

CIF 352\*288:

```
SCCB_salve_Address = 0x42;
write_SCCB(0x16,0x03);
write_SCCB(0x17,0xf9);
write_SCCB(0x18,0x5c);
write_SCCB(0x19,0x6c);
write_SCCB(0x1a,0x96);
write_SCCB(0x22,0x00);
write_SCCB(0xc8,0x01);
write_SCCB(0xc9,0x60);
write_SCCB(0xca,0x01);
```



```
write_SCCB(0xcb,0x20);
write_SCCB(0xcc,0x01);
write_SCCB(0xcd,0x60);
write_SCCB(0xce,0x01);
write_SCCB(0xcf,0x20);
```

QCIF 176\*144:

```
SCCB_salve_Address = 0x42;
write_SCCB(0x16,0x40);
write_SCCB(0x17,0x83);
write_SCCB(0x18,0x96);
write_SCCB(0x19,0x06);
write_SCCB(0x1a,0xf6);
write_SCCB(0x22,0x10);
write_SCCB(0xc8,0x02);
write_SCCB(0xc9,0x4b);
write_SCCB(0xca,0x00);
write_SCCB(0xcb,0xf0);
write_SCCB(0xcc,0x00);
write_SCCB(0xcd,0xb0);
write_SCCB(0xce,0x00);
write_SCCB(0xcf,0x90);
```

128\*96

```
:
SCCB_salve_Address = 0x42;
write_SCCB(0x16,0x03);
write_SCCB(0x17,0x69);
write_SCCB(0x18,0xa4);
write_SCCB(0x19,0x04);
write_SCCB(0x1a,0xf6);
write_SCCB(0x22,0x10);
write_SCCB(0xc8,0x02);
write_SCCB(0xc9,0x80);
write_SCCB(0xca,0x00);
write_SCCB(0xcb,0xf0);
write_SCCB(0xcc,0x00);
write_SCCB(0xcd,0x80);
write_SCCB(0xce,0x00);
write_SCCB(0xcf,0x60);
```

128\*200

```
:
SCCB_salve_Address = 0x42;
write_SCCB(0x16,0x03);
write_SCCB(0x17,0xed);
```

```

write_SCCB(0x18,0x62);
write_SCCB(0x19,0x04);
write_SCCB(0x1a,0xf6);
write_SCCB(0x22,0x10);
write_SCCB(0xc8,0x01);
write_SCCB(0xc9,0x78);
write_SCCB(0xca,0x00);
write_SCCB(0xcb,0xf0);
write_SCCB(0xcc,0x00);
write_SCCB(0xcd,0x80);
write_SCCB(0xce,0x00);
write_SCCB(0xcf,0xc8);

```

128\*128

:

```

SCCB_salve_Address = 0x42;
write_SCCB(0x16,0x03);
write_SCCB(0x17,0xb7);
write_SCCB(0x18,0x7d);
write_SCCB(0x19,0x04);
write_SCCB(0x1a,0xf6);
write_SCCB(0x22,0x10);
write_SCCB(0xc8,0x01);
write_SCCB(0xc9,0xe4);
write_SCCB(0xca,0x00);
write_SCCB(0xcb,0xf0);
write_SCCB(0xcc,0x00);
write_SCCB(0xcd,0x80);
write_SCCB(0xce,0x00);
write_SCCB(0xcf,0x80);

```

## Revision History

Rev1.00

Initial Release. Modified from OV7690.