OV5640 Camera Module

Software Application Notes

Last Modified: Nov. 15th, 2010

Document Revision: 1.1

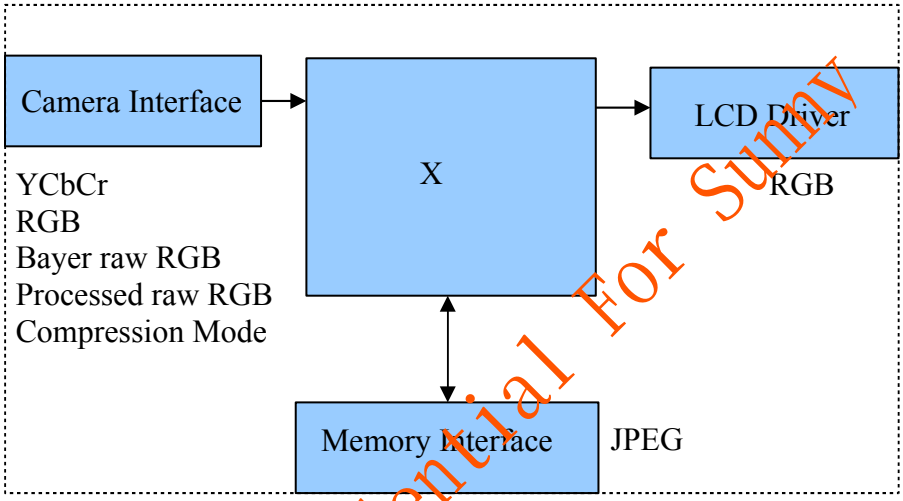
Table of Contents

NOTE: OV5640 use 16bits Register address and 8bits Register data.

1. How to Select Output format?

OV5640 support 5 output format: YcbCr422/420, RGB565/555/444, Bayer raw RGB, CCIR656, 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 CIP RAW over sensor 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 ,defect pixel correction 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.

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 OV5640 outputs only VGA format or SXGA format for preview and capture, ISP scaled VGA or SXGA image to other resolution that mobile device needed for LCD display and capture when image size below VGA or SXGA image . But for the image size bigger than SXGA format, OV5640 would output the size needed.

2.2 back-end without ISP

If back-end chip doesn't have image scale capability, then the LCD scaler of OV5640 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, OV5640 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, QXGA, QSXGA . For best quality, all capture size are all downscaled from QSXGA .

3. How to Adjust frame rate

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

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 15fps to 3.75fps. When environment is bright, the frame rate is increased to 15fps. When environment is dark, the frame rate is decreased to 3.75fps.

4.1 Night Mode VGA Preview with Fixed Frame Rate

3.75fps night mode for 60/50Hz light environment, 24Mhz clock input,6Mhz PCLK
i2c_salve_Address = 0x78;

```
write_I2c(0x3034 ,0x1a);
write_I2c(0x3035 ,0x61);
write_I2c(0x3036 ,0x46);
write_I2c(0x3037 ,0x13);
write_I2c(0x3038 ,0x00);
write_I2c(0x3039 ,0x00);
write_I2c(0x3a00 ,0x78);
write_I2c(0x3a08 ,0x01);
write_I2c(0x3a09 ,0x27);
write_I2c(0x3a0a ,0x00);
write_I2c(0x3a0b ,0xf6);
write_I2c(0x3a0d ,0x04);
write_I2c(0x3a0e ,0x04);
write_I2c(0x3a02 ,0x03);
write_I2c(0x3a03 ,0xd8);
write_I2c(0x3a14 ,0x03);
write_I2c(0x3a15 ,0xd8);
```

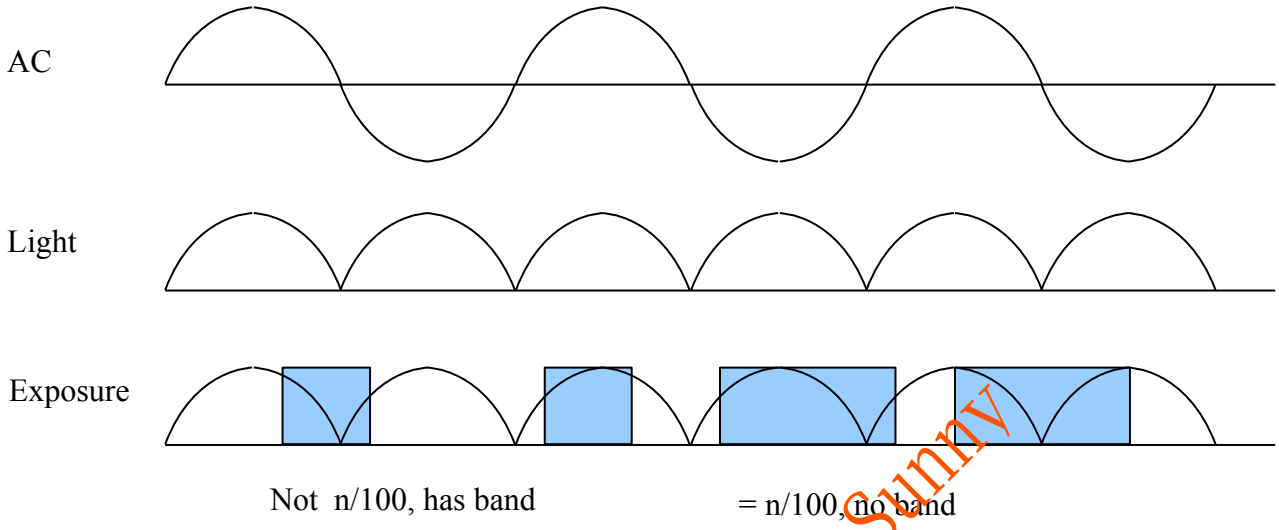
4.2 Night Mode VGA preview with Auto Frame Rate

15fps ~ 3.75fps night mode for 60/50Hz light environment, 24Mhz clock input,24Mhz PCLK

```
i2c_salve_Address = 0x78;
write_I2c(0x3034 ,0x1a);
write_I2c(0x3035 ,0x21);
write_I2c(0x3036 ,0x46);
write_I2c(0x3037 ,0x13);
write_I2c(0x3038 ,0x00);
write_I2c(0x3039 ,0x00);
write_I2c(0x3a00 ,0x7c);
write_I2c(0x3a08 ,0x01);
write_I2c(0x3a09 ,0x27);
write_I2c(0x3a0a ,0x00);
write_I2c(0x3a0b ,0xf6);
write_I2c(0x3a0d ,0x04);
write_I2c(0x3a0e ,0x04);
write_I2c(0x3a02 ,0x0b);
write_I2c(0x3a03 ,0x88);
write_I2c(0x3a14 ,0x0b);
write_I2c(0x3a15 ,0x88);
```

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 OV5640 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 15fps VGA preview, 24Mhz input clock
i2c_salve_Address = 0x78;
write_i2c(0x3c00, 0x00); bit[2]select 50/60hz banding, 0:50hz
write_i2c(0x3c01, 0x80); bit[7] banding filter Auto Detection on/off, 1 off
write_i2c(0x3a08, 0x01); //50Hz banding filter value 8 MSB
write_i2c(0x3a09, 0x27); //50Hz banding filter value 8 LSB
write_i2c(0x3a0a, 0x00); //60Hz banding filter value 8MSB
write_i2c(0x3a0b, 0xf6); //60Hz banding filter value 8 LSB
write_i2c(0x3a0e, 0x04); 50Hz maximum banding step
write_i2c(0x3a0d, 0x06); 60Hz maximum banding step

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. OV5640 could automatic select 50Hz or 60Hz banding filter based on light frequency detection.

QSXGA and any size DCW from QSXGA

```
i2c_salve_Address = 0x78;
write_I2c(0x3622 ,0x01);
write_I2c(0x3635 ,0x1c);
write_I2c(0x3634 ,0x40);
write_I2c(0x3c01 ,0x34);
write_I2c(0x3c00 ,0x00);
write_I2c(0x3c04 ,0x28);
write_I2c(0x3c05 ,0x98);
write_I2c(0x3c06 ,0x00);
write_I2c(0x3c07 ,0x08);
write_I2c(0x3c08 ,0x00);
write_I2c(0x3c09 ,0x1c);
write_I2c(0x300c ,0x22);
write_I2c(0x3c0a ,0x9c);
write_I2c(0x3c0b ,0x40);
```

5.5 Remove Light Band In Capture

Refer to 14.

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 in this conditions could not be removed for all CMOS sensors, not only OV5640. So there is no way to remove light band in this condition.

6. White Balance

OV5640 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
i2c_salve_Address = 0x78;
write_i2c(0x5183, 0x80); // Simple AWB, 0 for advanced 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,5M advanced AWB is selected.

7. Defect Pixel Correction

Defect pixel includes 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.
OV5640 has built-in defect pixel correction function. If OV5640 output YCbCr, RGB565, CIP raw RGB, the defect pixel correction function could be enabled to fix defect pixels. But if sensor Bayer raw RGB is used, the defect pixel correction function of sensor could not be used.
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 OV5640.
Settings
i2c_salve_Address = 0x78;
write_i2c(0x5000, 0x06); // Pixel Correction ON,bit[2:1]: 11,select enable

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 OV5640. 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. OV5640 can support 1080P for Video mode.

10. Digital zoom

If OV5640 output image smaller than XGA, it may support continuous digital zoom. For example

QSXGA	no digital zoom supported
XGA	1-2x
VGA	1-4x
QVGA	1-8x

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

11. OV5640 Functions

11.1 Light Mode

Advanced AWB

Advanced AWB

```
write_I2c(0x3406 ,0x00);
write_I2c(0x5192 ,0x04);
write_I2c(0x5191 ,0xf8);
write_I2c(0x5193 ,0x70);
write_I2c(0x5194 ,0xf0);
write_I2c(0x5195 ,0xf0);
write_I2c(0x518d ,0x3d);
write_I2c(0x518f ,0x54);
write_I2c(0x518e ,0x3d);
write_I2c(0x5190 ,0x54);
write_I2c(0x518b ,0xa8);
write_I2c(0x518c ,0xa8);
write_I2c(0x5187 ,0x18);
write_I2c(0x5188 ,0x18);
write_I2c(0x5189 ,0x6e);
write_I2c(0x518a ,0x68);
write_I2c(0x5186 ,0x1c);
write_I2c(0x5181 ,0x50);
write_I2c(0x5184 ,0x25);
write_I2c(0x5182 ,0x11);
write_I2c(0x5183 ,0x14);
write_I2c(0x5184 ,0x25);
write_I2c(0x5185 ,0x24);
```

Simple AWB
write_i2c(0x3406 ,0x 0),
write_i2c(0x5183 ,0x94),
write_i2c(0x5191 ,0xff),
write_i2c(0x5192 ,0x00),

Manual day

Manual day
write_i2c(0x3406 ,0x1),
write_i2c(0x3400 ,0x6),
write_i2c(0x3401 ,0x1c),
write_i2c(0x3402 ,0x4),
write_i2c(0x3403 ,0x0),
write_i2c(0x3404 ,0x4),
write_i2c(0x3405 ,0xf3),

Manual A

Manual A
write_i2c(0x3406 ,0x1),
write_i2c(0x3400 ,0x4),
write_i2c(0x3401 ,0x10),
write_i2c(0x3402 ,0x4),
write_i2c(0x3403 ,0x0),
write_i2c(0x3404 ,0x8),
write_i2c(0x3405 ,0xb6),

Manual cwf

Manual cwf
write_i2c(0x3406 ,0x1),
write_i2c(0x3400 ,0x5),
write_i2c(0x3401 ,0x48),
write_i2c(0x3402 ,0x4),
write_i2c(0x3403 ,0x0),
write_i2c(0x3404 ,0x7),
write_i2c(0x3405 ,0xcf),

Manual cloudy

Manual cloudy
write_i2c(0x3406 ,0x1),
write_i2c(0x3400 ,0x6),
write_i2c(0x3401 ,0x48),
write_i2c(0x3402 ,0x4),
write_i2c(0x3403 ,0x0),
write_i2c(0x3404 ,0x4),

write_i2c(0x3405 ,0xd3),

11.2 Color Saturation

The color saturation of OV5640 could be adjusted. High color saturation would make the picture looks more vivid, but the side effect is the bigger noise and not accurate skin color.

Saturation + 4
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x80)
write_i2c(0x5584 ,0x80)
write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

Saturation + 4

Saturation + 3
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x70)
write_i2c(0x5584 ,0x70)
write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

Saturation + 3

Saturation + 2
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x60)
write_i2c(0x5584 ,0x60)
write_i2c(0x5580 ,0x02)

Saturation + 2

write_i2c(0x5588 ,0x41)

Saturation + 1
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x50)
write_i2c(0x5584 ,0x50)
write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

Saturation + 1

Saturation 0
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x40)
write_i2c(0x5584 ,0x40)
write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

Saturation 0

Saturation -1
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x30)
write_i2c(0x5584 ,0x30)
write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

Saturation -1

Saturation – 2
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x20)
write_i2c(0x5584 ,0x20)

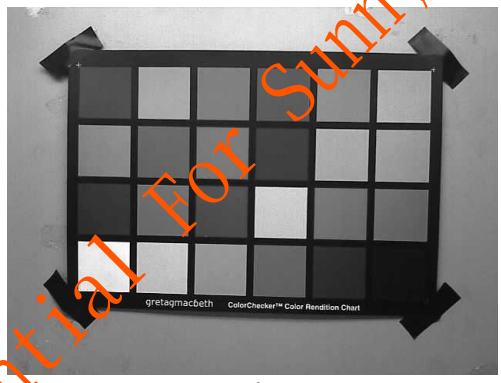
Saturation – 2

write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

Saturation -3
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x10)
write_i2c(0x5584 ,0x10)
write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

Saturation -3

Saturation – 4
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5583 ,0x00)
write_i2c(0x5584 ,0x00)
write_i2c(0x5580 ,0x02)
write_i2c(0x5588 ,0x41)

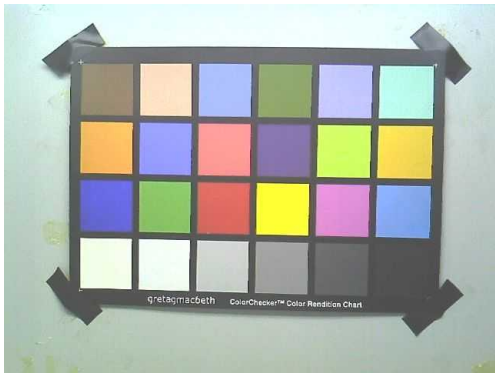


Saturation – 4

11.3 Brightness

The brightness of OV5640 could be adjusted. Higher brightness will make the picture more bright. The side effect of higher brightness is the picture looks foggy.

Brightness +4
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x40)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x01)



Brightness +4

Brightness +3
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x30)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x01)

Brightness +3

Brightness +2
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x20)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x01)

Brightness +2

Brightness +1
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x10)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x01)

Brightness 0
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x00)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x01)

Brightness 0

Brightness -1
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x10)

Brightness -1

write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x09)

Brightness -2

Brightness -2
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x20)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x09)

Brightness -3

Brightness -3
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587,0x30)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x09)

Brightness -4

Brightness -4
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5587 ,0x40)
write_i2c(0x5580 ,0x04)
write_i2c(0x5588 ,0x09)

11.4 Contrast

The contrast of OV5640 could be adjusted. Higher contrast will make the picture sharp. But the side effect is loosing dynamic range.

Contrast +4

Contrast +4
i2c_salve_Address = 0x78;

write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x30)
write_i2c(0x5585 ,0x30)
write_i2c(0x5588 ,0x41)

Contrast +3
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x2c)
write_i2c(0x5585 ,0x2c)
write_i2c(0x5588 ,0x41)

Contrast +2
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x28)
write_i2c(0x5585 ,0x28)
write_i2c(0x5588 ,0x41)

Contrast +1

Contrast +1
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x24)
write_i2c(0x5585 ,0x24)
write_i2c(0x5588 ,0x41)

Contrast 0
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x20)

Contrast -1

write_i2c(0x5585 ,0x20)
write_i2c(0x5588 ,0x41)

Contrast -1
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x1c)
write_i2c(0x5585 ,0x1c)
write_i2c(0x5588 ,0x41)

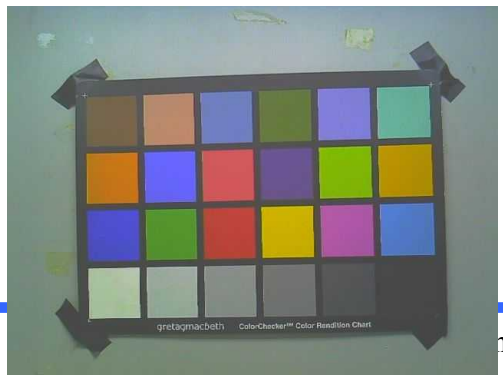
Contrast -2
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x18)
write_i2c(0x5585 ,0x18)
write_i2c(0x5588 ,0x41)

Contrast -2

Contrast -3
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586 ,0x14)
write_i2c(0x5585 ,0x14)
write_i2c(0x5588 ,0x41)

Contrast -3

Contrast -4
i2c_salve_Address = 0x78;
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x04)
write_i2c(0x5586,0x10)
write_i2c(0x5585 ,0x10)
write_i2c(0x5588 ,0x41)



Contrast -4

11.5 Hue

OV5640 support Hue tuning.

i2c_salve_Address = 0x78;

-180 degree

```
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x80)
write_i2c(0x5582 ,0x00)
write_i2c(0x5588 ,0x32)
```

-180 degree

-150 degree

```
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x6f)
write_i2c(0x5582 ,0x40)
write_i2c(0x5588 ,0x32)
```

-150 degree

-120 degree

```
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x40)
write_i2c(0x5582 ,0x6f)
write_i2c(0x5588 ,0x32)
```

-120 degree

-90 degree

```
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x00)
write_i2c(0x5582 ,0x80)
write_i2c(0x5588 ,0x02)
```

-90 degree

-60 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x40)
write_i2c(0x5582 ,0x6f)
write_i2c(0x5588 ,0x02)

-60 degree

-30 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x6f)
write_i2c(0x5582 ,0x40)
write_i2c(0x5588 ,0x02)

+0 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x80)
write_i2c(0x5582 ,0x00)
write_i2c(0x5588 ,0x01)

+0 degree

+30 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x6f)
write_i2c(0x5582 ,0x40)
write_i2c(0x5588 ,0x01)

+30 degree

+60 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x40)
write_i2c(0x5582 ,0x6f)
write_i2c(0x5588 ,0x01)

+60 degree

+90 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x00)
write_i2c(0x5582 ,0x80)
write_i2c(0x5588 ,0x31)

+90 degree

+120 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x40)
write_i2c(0x5582 ,0x6f)
write_i2c(0x5588 ,0x31)

+120 degree

+150 degree
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x01)
write_i2c(0x5581 ,0x6f)
write_i2c(0x5582 ,0x40)
write_i2c(0x5588 ,0x31)

+150 degree

11.6 Special effects

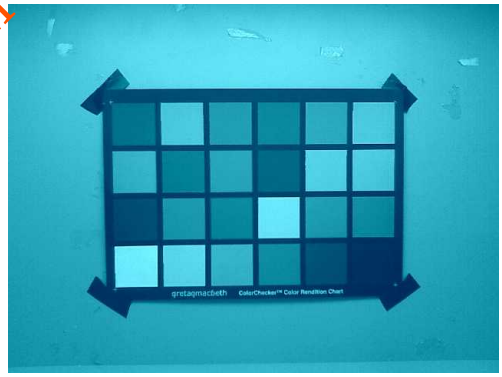
OV5640 support some special effects such as B/W, negative, sepia, bluish, reddish, greenish,negative, etc. If users need other special effects, it should be supported by back-end chips.
i2c_salve_Address = 0x78;

Normal
write_i2c(0x5001 ,0x7f)
write_i2c(0x5580 ,0x00)

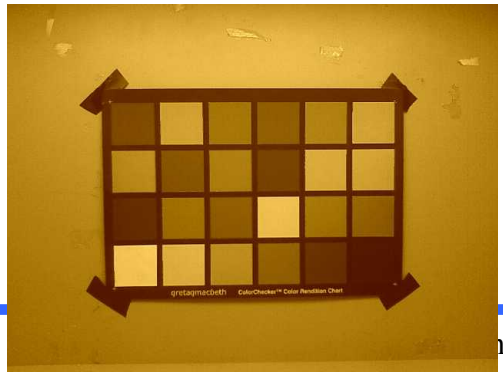
B&W
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x18)
write_i2c(0x5583 ,0x80)
write_i2c(0x5584 ,0x80)

B&W

Bluish
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x18)
write_i2c(0x5583 ,0xa0)
write_i2c(0x5584,0x40)


Bluish

Sepia
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x18)
write_i2c(0x5583 ,0x40)
write_i2c(0x5584 ,0xa0)


Sepia

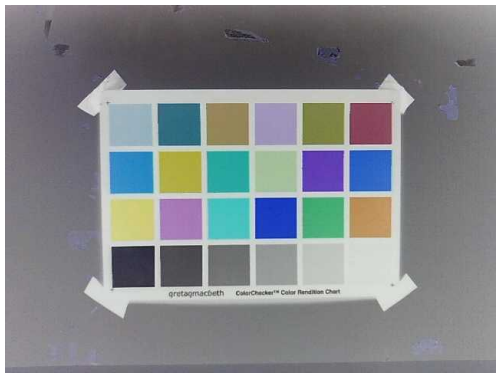
Reddish
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x18)
write_i2c(0x5583 ,0x80)
write_i2c(0x5584 ,0xc0)

Reddish

Greenish
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x18)
write_i2c(0x5583 ,0x60)
write_i2c(0x5584 ,0x60)

Greenish

Negative
write_i2c(0x5001 ,0xff)
write_i2c(0x5580 ,0x40)



Negative

11.7 Exposure level

OV5640 support different exposure level. It can increase/decrease target brightness by change exposure/gain auto. OV5640 support Average algorithm.

i2c_salve_Address = 0x78;

-1.7EV
write_i2c(0x3a0f ,0x10)
write_i2c(0x3a10 ,0x08)
write_i2c(0x3a1b ,0x10)
write_i2c(0x3a1e ,0x08)
write_i2c(0x3a11 ,0x20)
write_i2c(0x3a1f ,0x10)

-1.7EV

-1.3EV
write_i2c(0x3a0f ,0x18)
write_i2c(0x3a10 ,0x10)
write_i2c(0x3a1b ,0x18)
write_i2c(0x3a1e ,0x10)
write_i2c(0x3a11 ,0x30)
write_i2c(0x3a1f ,0x10)

-1.3EV

-1.0EV
write_i2c(0x3a0f ,0x20)
write_i2c(0x3a10 ,0x18)
write_i2c(0x3a11 ,0x41)
write_i2c(0x3a1b ,0x20)
write_i2c(0x3a1e ,0x18)
write_i2c(0x3a1f ,0x10)

-1.0EV

-0.7EV
write_i2c(0x3a0f ,0x28)
write_i2c(0x3a10 ,0x20)
write_i2c(0x3a11 ,0x51)

-0.7EV

write_i2c(0x3a1b ,0x28)
write_i2c(0x3a1e ,0x20)
write_i2c(0x3a1f ,0x10)

-0.3EV

-0.3EV
write_i2c(0x3a0f ,0x30)
write_i2c(0x3a10 ,0x28)
write_i2c(0x3a11 ,0x61)
write_i2c(0x3a1b ,0x30)
write_i2c(0x3a1e ,0x28)
write_i2c(0x3a1f ,0x10)

default

default
write_i2c(0x3a0f ,0x38)
write_i2c(0x3a10 ,0x30)
write_i2c(0x3a11 ,0x61)
write_i2c(0x3a1b ,0x38)
write_i2c(0x3a1e ,0x30)
write_i2c(0x3a1f ,0x10)

0.3EV

write_i2c(0x3a0f ,0x40)
write_i2c(0x3a10 ,0x38)
write_i2c(0x3a11 ,0x71)
write_i2c(0x3a1b ,0x40)
write_i2c(0x3a1e ,0x38)
write_i2c(0x3a1f ,0x10)

0.7EV

0.7EV
write_i2c(0x3a0f ,0x48)
write_i2c(0x3a10 ,0x40)

write_i2c(0x3a11 ,0x80)
write_i2c(0x3a1b ,0x48)
write_i2c(0x3a1e ,0x40)
write_i2c(0x3a1f ,0x20)

1.0EV

1.0EV
write_i2c(0x3a0f ,0x50)
write_i2c(0x3a10 ,0x48)
write_i2c(0x3a11 ,0x90)
write_i2c(0x3a1b ,0x50)
write_i2c(0x3a1e ,0x48)
write_i2c(0x3a1f ,0x20)

1.3EV

1.3EV
write_i2c(0x3a0f ,0x58)
write_i2c(0x3a10 ,0x50)
write_i2c(0x3a11 ,0x91)
write_i2c(0x3a1b ,0x58)
write_i2c(0x3a1e ,0x50)
write_i2c(0x3a1f ,0x20)

1.7EV

1.7EV
write_i2c(0x3a0f ,0x60)
write_i2c(0x3a10 ,0x58)
write_i2c(0x3a11 ,0xa0)
write_i2c(0x3a1b ,0x60)
write_i2c(0x3a1e ,0x58)
write_i2c(0x3a1f ,0x20)

11.8 Sharpness

i2c_salve_Address = 0x78;

Sharpness OFF

Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x00);

Sharpness 1
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x02);

Sharpness 2
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x04);

Sharpness 3
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x08);

Sharpness 4
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x0c);

Sharpness 5
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x10);

Sharpness 6
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x14);

Sharpness 7
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x18);

Sharpness 8
Write_i2c(0x5308 ,0x65);
Write_i2c(0x5302 ,0x20);

Sharpness Auto
Write_i2c(0x5308 ,0x25);
Write_i2c(0x5300 ,0x08);
Write_i2c(0x5301 ,0x30);
Write_i2c(0x5302 ,0x10);
Write_i2c(0x5303 ,0x00);
Write_i2c(0x5309 ,0x08);
Write_i2c(0x530a ,0x30);
Write_i2c(0x530b ,0x04);
Write_i2c(0x530c ,0x06);

MIRROR

11.9 Mirror/Flip

i2c_salve_Address = 0x78;

MIRROR

```
reg3820 = read_i2c(0x3820)
reg3820 = reg3820 & 0xf9
reg3820 = reg3820 | 0x00
write_i2c(0x3820, reg3820 )
reg3821 = read_i2c(0x3821)
reg3821 = reg3821 & 0xf9
reg3821 = reg3821 | 0x06
write_i2c(0x3821, reg3821 )
```

FLIP

FLIP

```
reg3820 = read_i2c(0x3820)
reg3820 = reg3820 & 0xf9
reg3820 = reg3820 | 0x06
write_i2c(0x3820, reg3820 )
reg3821 = read_i2c(0x3821)
reg3821 = reg3821 & 0xf9
reg3821 = reg3821 | 0x00
write_i2c(0x3821, reg3821 )
```

MIRROR&FLIP

MIRROR&FLIP

```
reg3820 = read_i2c(0x3820)
reg3820 = reg3820 & 0xf9
reg3820 = reg3820 | 0x06
write_i2c(0x3820, reg3820 )
reg3821 = read_i2c(0x3821)
reg3821 = reg3821 & 0xf9
reg3821 = reg3821 | 0x06
write_i2c(0x3821, reg3821 )
```


Normal

```
reg3820 = read_i2c(0x3820)
reg3820 = reg3820 & 0xf9
reg3820 = reg3820 | 0x00
write_i2c(0x3820, reg3820 )
reg3821 = read_i2c(0x3821)
reg3821 = reg3821 & 0xf9
reg3821 = reg3821 | 0x00
write_i2c(0x3821, reg3821 )
```

NORML

11.10 YUV Sequence

0x4300[0:1] control YUV sequence
Y U Y V
write_i2c(0x4300, 0x20)

Y V Y U
write_i2c(0x4300, 0x21)

V Y U Y
write_i2c(0x4300, 0x23)

U Y V Y
write_i2c(0x4300, 0x22)

11.11 Clock Polarity

Data valid VSYNC high
0x4740[0] Control VSYNC polarity
1: Data valid VSYNC High
0: Data valid VSYNC low

0x4740[5] Control PCLK polarity
1: Data update at Falling-edge
0: Data update at Rising-edge

0x4740[1] Control HREF polarity
0:Data valid HREF high
1: Data valid HREF Low

11.12 Compress quality

Register 0x4407[5:0] is for the compress quality

High quality :

write_i2c(0x4407, 0x02)

default quality :

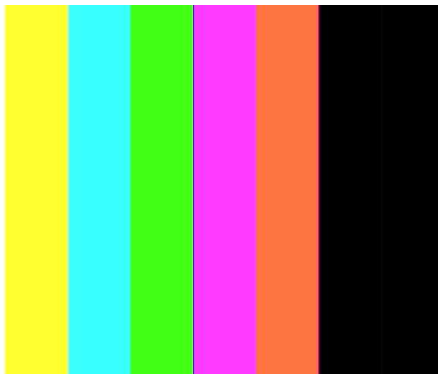
write_i2c(0x4407, 0x04)

low quality :

write_i2c(0x4407, 0x08)

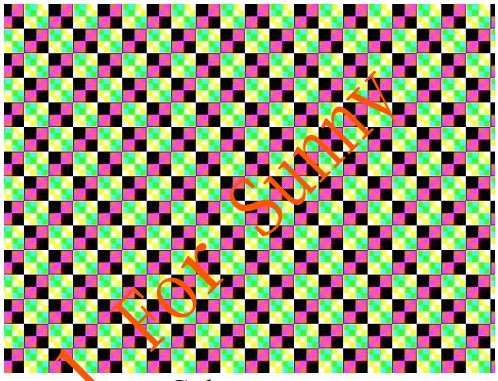
11.13 Test Pattern

Color bar
write_i2c(0x503d , 0x80)
write_i2c(0x4741, 0x00)



Color bar

Color square
write_i2c(0x503d , 0x82)
write_i2c(0x4741, 0x0)



Color square

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 OV5640 could be turned on to compensate the corner brightness and make the whole picture looks same bright.
settings
i2c_salve_Address = 0x78;
write_i2c(0x5000,0xff);bit[7]: enable lens correction

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 OV5640 the focus distance is about 140~160cm. The depth of field is about from 70~80cm

to infinite. If checking resolution of camera module, the resolution chart should be placed 140~160 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 foggy. 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

OV5640 output maximum 15fps QSXGA and maximum 30fps 1080P(1920*1080). Resolution from 1080P can reach up maximum 30fps, resolution from QSXGA can reach up maximum 15fps.

13.1.1 VGA Preview

```
; for the setting , 24M Mlck input and 24M Plck output
;15fps YUV mode
write_i2c(0x3008,0x82);
write_i2c(0x3103,0x03);
write_i2c(0x3017,0xff);
write_i2c(0x3018,0xff);
write_i2c(0x3108,0x01);
write_i2c(0x3037,0x13);
write_i2c(0x3630,0x2e);
write_i2c(0x3632,0xe2);
write_i2c(0x3633,0x23);
write_i2c(0x3634,0x44);
write_i2c(0x3621,0xe0);
write_i2c(0x3704,0xa0);
write_i2c(0x3703,0x5a);
write_i2c(0x3715,0x78);
write_i2c(0x3717,0x01);
```

```
write_i2c(0x370b,0x60);
write_i2c(0x3705,0x1a);
write_i2c(0x3905,0x02);
write_i2c(0x3906,0x10);
write_i2c(0x3901,0x0a);
write_i2c(0x3731,0x12);
write_i2c(0x3600,0x08);
write_i2c(0x3601,0x33);
write_i2c(0x471c,0x50);
write_i2c(0x3820,0x41);
write_i2c(0x3821,0x07);
write_i2c(0x3814,0x31);
write_i2c(0x3815,0x31);
write_i2c(0x3800,0x00);
write_i2c(0x3801,0x00);
write_i2c(0x3802,0x00);
write_i2c(0x3803,0x04);
write_i2c(0x3804,0x0a);
write_i2c(0x3805,0x3f);
write_i2c(0x3806,0x07);
write_i2c(0x3807,0x9b);
write_i2c(0x3808,0x02);
write_i2c(0x3809,0x80);
write_i2c(0x380a,0x01);
write_i2c(0x380b,0xe0);
write_i2c(0x380c,0x07);
write_i2c(0x380d,0x68);
write_i2c(0x380e,0x03);
write_i2c(0x380f,0xd8);
write_i2c(0x3810,0x00);
write_i2c(0x3811,0x10);
write_i2c(0x3812,0x00);
write_i2c(0x3813,0x06);
write_i2c(0x3618,0x00);
write_i2c(0x3612,0x49);
write_i2c(0x3708,0x62);
write_i2c(0x3709,0x52);
write_i2c(0x370c,0x03);
write_i2c(0x3a02,0x03);
write_i2c(0x3a03,0xd8);
write_i2c(0x3a08,0x01);
write_i2c(0x3a09,0x27);
write_i2c(0x3a0a,0x00);
write_i2c(0x3a0b,0xf6);
write_i2c(0x3a0e,0x03);
write_i2c(0x3a0d,0x04);
write_i2c(0x3a14,0x03);
write_i2c(0x3a15,0xd8);
```

```
write_i2c(0x4001,0x02);
write_i2c(0x4004,0x02);
write_i2c(0x3002,0x1c);
write_i2c(0x3006,0xc3);
write_i2c(0x4300,0x30);
write_i2c(0x501f,0x00);
write_i2c(0x4713,0x03);
write_i2c(0x3035,0x11);
write_i2c(0x3036,0x46);
write_i2c(0x4407,0x04);
write_i2c(0x460b,0x35);
write_i2c(0x460c,0x22);
write_i2c(0x3824,0x02);
write_i2c(0x5000,0xa7);
write_i2c(0x5001,0xa3);
write_i2c(0x5000,0xa7);
write_i2c(0x3622,0x01);
write_i2c(0x3635,0x1c);
write_i2c(0x3634,0x40);
write_i2c(0x3c01,0x34);
write_i2c(0x3c00,0x00);
write_i2c(0x3c04,0x28);
write_i2c(0x3c05,0x98);
write_i2c(0x3c06,0x00);
write_i2c(0x3c07,0x08);
write_i2c(0x3c08,0x00);
write_i2c(0x3c09,0x1c);
write_i2c(0x300c,0x22);
write_i2c(0x3c0a,0x9c);
write_i2c(0x3c0b,0x40);
write_i2c(0x5180,0xff);
write_i2c(0x5181,0xf2);
write_i2c(0x5182,0x00);
write_i2c(0x5183,0x94);
write_i2c(0x5184,0x25);
write_i2c(0x5185,0x24);
write_i2c(0x5186,0x06);
write_i2c(0x5187,0x08);
write_i2c(0x5188,0x08);
write_i2c(0x5189,0x78);
write_i2c(0x518a,0x54);
write_i2c(0x518b,0xb2);
write_i2c(0x518c,0xb2);
write_i2c(0x518d,0x44);
write_i2c(0x518e,0x3d);
write_i2c(0x518f,0x58);
write_i2c(0x5190,0x46);
write_i2c(0x5191,0xf8);
```

```
write_i2c(0x5192,0x04);
write_i2c(0x5193,0x70);
write_i2c(0x5194,0xf0);
write_i2c(0x5195,0xf0);
write_i2c(0x5196,0x03);
write_i2c(0x5197,0x01);
write_i2c(0x5198,0x04);
write_i2c(0x5199,0x12);
write_i2c(0x519a,0x04);
write_i2c(0x519b,0x00);
write_i2c(0x519c,0x06);
write_i2c(0x519d,0x82);
write_i2c(0x519e,0x38);
write_i2c(0x5381,0x1c);
write_i2c(0x5382,0x5a);
write_i2c(0x5383,0x06);
write_i2c(0x5384,0x20);
write_i2c(0x5385,0x80);
write_i2c(0x5386,0xa0);
write_i2c(0x5387,0xa2);
write_i2c(0x5388,0xa0);
write_i2c(0x5389,0x02);
write_i2c(0x538a,0x01);
write_i2c(0x538b,0x98);
write_i2c(0x5300,0x08);
write_i2c(0x5301,0x30);
write_i2c(0x5302,0x10);
write_i2c(0x5303,0x00);
write_i2c(0x5304,0x08);
write_i2c(0x5305,0x30);
write_i2c(0x5306,0x08);
write_i2c(0x5307,0x16);
write_i2c(0x5309,0x08);
write_i2c(0x530a,0x30);
write_i2c(0x530b,0x04);
write_i2c(0x530c,0x06);
write_i2c(0x5480,0x01);
write_i2c(0x5481,0x08);
write_i2c(0x5482,0x14);
write_i2c(0x5483,0x28);
write_i2c(0x5484,0x51);
write_i2c(0x5485,0x65);
write_i2c(0x5486,0x71);
write_i2c(0x5487,0x7d);
write_i2c(0x5488,0x87);
write_i2c(0x5489,0x91);
write_i2c(0x548a,0x9a);
write_i2c(0x548b,0xaa);
```

```
write_i2c(0x548c,0xb8);
write_i2c(0x548d,0xcd);
write_i2c(0x548e,0xdd);
write_i2c(0x548f,0xea);
write_i2c(0x5490,0x1d);
write_i2c(0x5580,0x02);
write_i2c(0x5583,0x40);
write_i2c(0x5584,0x10);
write_i2c(0x5589,0x10);
write_i2c(0x558a,0x00);
write_i2c(0x558b,0xf8);
write_i2c(0x5800,0x23);
write_i2c(0x5801,0x15);
write_i2c(0x5802,0x10);
write_i2c(0x5803,0x10);
write_i2c(0x5804,0x15);
write_i2c(0x5805,0x23);
write_i2c(0x5806,0x0c);
write_i2c(0x5807,0x08);
write_i2c(0x5808,0x05);
write_i2c(0x5809,0x05);
write_i2c(0x580a,0x08);
write_i2c(0x580b,0x0c);
write_i2c(0x580c,0x07);
write_i2c(0x580d,0x03);
write_i2c(0x580e,0x00);
write_i2c(0x580f,0x00);
write_i2c(0x5810,0x03);
write_i2c(0x5811,0x07);
write_i2c(0x5812,0x07);
write_i2c(0x5813,0x03);
write_i2c(0x5814,0x00);
write_i2c(0x5815,0x00);
write_i2c(0x5816,0x03);
write_i2c(0x5817,0x07);
write_i2c(0x5818,0x0b);
write_i2c(0x5819,0x08);
write_i2c(0x581a,0x05);
write_i2c(0x581b,0x05);
write_i2c(0x581c,0x07);
write_i2c(0x581d,0x0b);
write_i2c(0x581e,0x2a);
write_i2c(0x581f,0x16);
write_i2c(0x5820,0x11);
write_i2c(0x5821,0x11);
write_i2c(0x5822,0x15);
write_i2c(0x5823,0x29);
write_i2c(0x5824,0xbf);
```

```
write_i2c(0x5825,0xaf);
write_i2c(0x5826,0x9f);
write_i2c(0x5827,0xaf);
write_i2c(0x5828,0xdf);
write_i2c(0x5829,0x6f);
write_i2c(0x582a,0x8e);
write_i2c(0x582b,0xab);
write_i2c(0x582c,0x9e);
write_i2c(0x582d,0x7f);
write_i2c(0x582e,0x4f);
write_i2c(0x582f,0x89);
write_i2c(0x5830,0x86);
write_i2c(0x5831,0x98);
write_i2c(0x5832,0x6f);
write_i2c(0x5833,0x4f);
write_i2c(0x5834,0x6e);
write_i2c(0x5835,0x7b);
write_i2c(0x5836,0x7e);
write_i2c(0x5837,0x6f);
write_i2c(0x5838,0xde);
write_i2c(0x5839,0xbf);
write_i2c(0x583a,0x9f);
write_i2c(0x583b,0xbf);
write_i2c(0x583c,0xec);
write_i2c(0x5025,0x00);
write_i2c(0x3a0f,0x30);
write_i2c(0x3a10,0x28);
write_i2c(0x3a1b,0x30);
write_i2c(0x3a1e,0x26);
write_i2c(0x3a11,0x60);
write_i2c(0x3a1f,0x14);
write_i2c(0x3a18,0x00);
write_i2c(0x3a19,0xf8);
write_i2c(0x3035,0x21);
```


13.1.2 Other size preview

to vga

```
write_i2c(0x3800 ,0x0 );
write_i2c(0x3801 ,0x0 );
write_i2c(0x3802 ,0x0 );
write_i2c(0x3803 ,0x4 );
write_i2c(0x3804 ,0xa );
```

```
write_i2c(0x3805 ,0x3f);
write_i2c(0x3806 ,0x7 );
write_i2c(0x3807 ,0x9b);
write_i2c(0x3808 ,0x2 );
write_i2c(0x3809 ,0x80);
write_i2c(0x380a ,0x1 );
write_i2c(0x380b ,0xe0);
write_i2c(0x380c ,0x7 );
write_i2c(0x380d ,0x68);
write_i2c(0x380e ,0x3 );
write_i2c(0x380f ,0xd8);
```

to qvga

```
write_i2c(0x3800 ,0x0 );
write_i2c(0x3801 ,0x0 );
write_i2c(0x3802 ,0x0 );
write_i2c(0x3803 ,0x4 );
write_i2c(0x3804 ,0xa );
write_i2c(0x3805 ,0x3f);
write_i2c(0x3806 ,0x7 );
write_i2c(0x3807 ,0x9b);
write_i2c(0x3808 ,0x1 );
write_i2c(0x3809 ,0x40);
write_i2c(0x380a ,0x0 );
write_i2c(0x380b ,0xf0);
write_i2c(0x380c ,0x7 );
write_i2c(0x380d ,0x68);
write_i2c(0x380e ,0x3 );
write_i2c(0x380f ,0xd8);
```

to svga

write_i2c(0x3800 ,0x0);
write_i2c(0x3801 ,0x0);
write_i2c(0x3802 ,0x0);
write_i2c(0x3803 ,0x4);
write_i2c(0x3804 ,0xa);
write_i2c(0x3805 ,0x3f);
write_i2c(0x3806 ,0x7);
write_i2c(0x3807 ,0x9b);
write_i2c(0x3808 ,0x3);
write_i2c(0x3809 ,0x20);
write_i2c(0x380a ,0x2);
write_i2c(0x380b ,0x58);
write_i2c(0x380c ,0x7);
write_i2c(0x380d ,0x68);
write_i2c(0x380e ,0x3);

to cif

write_i2c(0x3800 ,0x0);
write_i2c(0x3801 ,0x0);
write_i2c(0x3802 ,0x0);
write_i2c(0x3803 ,0x4);
write_i2c(0x3804 ,0xa);
write_i2c(0x3805 ,0x3f);
write_i2c(0x3806 ,0x7);
write_i2c(0x3807 ,0x9b);
write_i2c(0x3808 ,0x1);
write_i2c(0x3809 ,0x60);
write_i2c(0x380a ,0x1);
write_i2c(0x380b ,0x20);
write_i2c(0x380c ,0x7);
write_i2c(0x380d ,0x68);

write_i2c(0x380e ,0x3);

13.1.3 VGA 90fps video

write_i2c(0x3008 ,0x82);
write_i2c(0x3103 ,0x03);
write_i2c(0x3017 ,0xff);
write_i2c(0x3018 ,0xff);
write_i2c(0x3108 ,0x01);
write_i2c(0x3037 ,0x13);
write_i2c(0x3630 ,0x2e);
write_i2c(0x3632 ,0xe2);
write_i2c(0x3633 ,0x23);
write_i2c(0x3634 ,0x44);
write_i2c(0x3621 ,0xe0);
write_i2c(0x3704 ,0xa0);
write_i2c(0x3703 ,0x5a);
write_i2c(0x3715 ,0x78);
write_i2c(0x3717 ,0x01);
write_i2c(0x370b ,0x60);
write_i2c(0x3705 ,0x1a);
write_i2c(0x3905 ,0x02);
write_i2c(0x3906 ,0x10);
write_i2c(0x3901 ,0x0a);
write_i2c(0x3731 ,0x12);
write_i2c(0x3600 ,0x08);
write_i2c(0x3601 ,0x33);
write_i2c(0x471c ,0x50);
write_i2c(0x3820 ,0x41);
write_i2c(0x3821 ,0x27);
write_i2c(0x3814 ,0x71);
write_i2c(0x3815 ,0x35);
write_i2c(0x3800 ,0x00);
write_i2c(0x3801 ,0x00);

```
write_i2c(0x3802 ,0x00);
write_i2c(0x3803 ,0x00);
write_i2c(0x3804 ,0x0a);
write_i2c(0x3805 ,0x3f);
write_i2c(0x3806 ,0x07);
write_i2c(0x3807 ,0x9f);
write_i2c(0x3808 ,0x02);
write_i2c(0x3809 ,0x80);
write_i2c(0x380a ,0x01);
write_i2c(0x380b ,0xe0);
write_i2c(0x380c ,0x07);
write_i2c(0x380d ,0x58);
write_i2c(0x380e ,0x01);
write_i2c(0x380f ,0xf0);
write_i2c(0x3810 ,0x00);
write_i2c(0x3811 ,0x08);
write_i2c(0x3812 ,0x00);
write_i2c(0x3813 ,0x02);
write_i2c(0x3618 ,0x00);
write_i2c(0x3612 ,0x49);
write_i2c(0x3708 ,0x66);
write_i2c(0x3709 ,0x52);
write_i2c(0x370c ,0x03);
write_i2c(0x3a02 ,0x01);
write_i2c(0x3a03 ,0xf0);
write_i2c(0x3a08 ,0x01);
write_i2c(0x3a09 ,0xbe);
write_i2c(0x3a0a ,0x01);
write_i2c(0x3a0b ,0x74);
write_i2c(0x3a0e ,0x01);
write_i2c(0x3a0d ,0x01);
write_i2c(0x3a14 ,0x01);
write_i2c(0x3a15 ,0xf0);
write_i2c(0x4001 ,0x02);
```

```
write_i2c(0x4004 ,0x02);
write_i2c(0x3000 ,0x00);
write_i2c(0x3002 ,0x00);
write_i2c(0x3004 ,0xff);
write_i2c(0x3006 ,0xff);
write_i2c(0x4300 ,0x30);
write_i2c(0x501f ,0x00);
write_i2c(0x4713 ,0x03);
write_i2c(0x3035 ,0x11);
write_i2c(0x4407 ,0x04);
write_i2c(0x460b ,0x35);
write_i2c(0x460c ,0x22);
write_i2c(0x3824 ,0x04);
write_i2c(0x5181 ,0xf2);
write_i2c(0x5182 ,0x00);
write_i2c(0x5197 ,0x01);
write_i2c(0x519e ,0x38);
write_i2c(0x5000 ,0xa7);
write_i2c(0x5001 ,0x83);
write_i2c(0x5000 ,0xa7);
write_i2c(0x3622 ,0x01);
write_i2c(0x3635 ,0x1c);
write_i2c(0x3634 ,0x40);
write_i2c(0x3c01 ,0x34);
write_i2c(0x3c00 ,0x00);
write_i2c(0x3c04 ,0x28);
write_i2c(0x3c05 ,0x98);
write_i2c(0x3c06 ,0x00);
write_i2c(0x3c07 ,0x08);
write_i2c(0x3c08 ,0x00);
write_i2c(0x3c09 ,0x1c);
write_i2c(0x300c ,0x22);
write_i2c(0x3c0a ,0x9c);
write_i2c(0x3c0b ,0x40);
```

write_i2c(0x5180 ,0xff);
write_i2c(0x5181 ,0xf2);
write_i2c(0x5182 ,0x00);
write_i2c(0x5183 ,0x14);
write_i2c(0x5184 ,0x25);
write_i2c(0x5185 ,0x24);
write_i2c(0x5186 ,0x06);
write_i2c(0x5187 ,0x09);
write_i2c(0x5188 ,0x09);
write_i2c(0x5189 ,0x7b);
write_i2c(0x518a ,0x54);
write_i2c(0x518b ,0xb8);
write_i2c(0x518c ,0xb2);
write_i2c(0x518d ,0x42);
write_i2c(0x518e ,0x3d);
write_i2c(0x518f ,0x56);
write_i2c(0x5190 ,0x46);
write_i2c(0x5191 ,0xf8);
write_i2c(0x5192 ,0x04);
write_i2c(0x5193 ,0x70);
write_i2c(0x5194 ,0xf0);
write_i2c(0x5195 ,0xf0);
write_i2c(0x5196 ,0x03);
write_i2c(0x5197 ,0x01);
write_i2c(0x5198 ,0x04);
write_i2c(0x5199 ,0x12);
write_i2c(0x519a ,0x04);
write_i2c(0x519b ,0x00);
write_i2c(0x519c ,0x06);
write_i2c(0x519d ,0x82);
write_i2c(0x519e ,0x38);
write_i2c(0x5381 ,0x1c);
write_i2c(0x5382 ,0x5a);
write_i2c(0x5383 ,0x06);

```
write_i2c(0x5384 ,0x1b);
write_i2c(0x5385 ,0x6d);
write_i2c(0x5386 ,0x88);
write_i2c(0x5387 ,0x8a);
write_i2c(0x5388 ,0x88);
write_i2c(0x5389 ,0x02);
write_i2c(0x538a ,0x01);
write_i2c(0x538b ,0x98);
write_i2c(0x5300 ,0x08);
write_i2c(0x5301 ,0x30);
write_i2c(0x5302 ,0x10);
write_i2c(0x5303 ,0x00);
write_i2c(0x5304 ,0x08);
write_i2c(0x5305 ,0x30);
write_i2c(0x5306 ,0x08);
write_i2c(0x5307 ,0x16);
write_i2c(0x5309 ,0x08);
write_i2c(0x530a ,0x30);
write_i2c(0x530b ,0x04);
write_i2c(0x530c ,0x06);
write_i2c(0x5480 ,0x01);
write_i2c(0x5481 ,0x08);
write_i2c(0x5482 ,0x14);
write_i2c(0x5483 ,0x28);
write_i2c(0x5484 ,0x51);
write_i2c(0x5485 ,0x65);
write_i2c(0x5486 ,0x71);
write_i2c(0x5487 ,0x7d);
write_i2c(0x5488 ,0x87);
write_i2c(0x5489 ,0x91);
write_i2c(0x548a ,0x9a);
write_i2c(0x548b ,0xaa);
write_i2c(0x548c ,0xb8);
write_i2c(0x548d ,0xcd);
```

write_i2c(0x548e ,0xdd);
write_i2c(0x548f ,0xea);
write_i2c(0x5490 ,0x1d);
write_i2c(0x5580 ,0x02);
write_i2c(0x5583 ,0x40);
write_i2c(0x5584 ,0x10);
write_i2c(0x5589 ,0x10);
write_i2c(0x558a ,0x00);
write_i2c(0x558b ,0xf8);
write_i2c(0x5800 ,0x23);
write_i2c(0x5801 ,0x15);
write_i2c(0x5802 ,0x10);
write_i2c(0x5803 ,0x10);
write_i2c(0x5804 ,0x15);
write_i2c(0x5805 ,0x23);
write_i2c(0x5806 ,0x0c);
write_i2c(0x5807 ,0x08);
write_i2c(0x5808 ,0x05);
write_i2c(0x5809 ,0x05);
write_i2c(0x580a ,0x08);
write_i2c(0x580b ,0x0c);
write_i2c(0x580c ,0x07);
write_i2c(0x580d ,0x03);
write_i2c(0x580e ,0x00);
write_i2c(0x580f ,0x00);
write_i2c(0x5810 ,0x03);
write_i2c(0x5811 ,0x07);
write_i2c(0x5812 ,0x07);
write_i2c(0x5813 ,0x03);
write_i2c(0x5814 ,0x00);
write_i2c(0x5815 ,0x00);
write_i2c(0x5816 ,0x03);
write_i2c(0x5817 ,0x07);
write_i2c(0x5818 ,0x0b);

write_i2c(0x5819 ,0x08);
write_i2c(0x581a ,0x05);
write_i2c(0x581b ,0x05);
write_i2c(0x581c ,0x07);
write_i2c(0x581d ,0x0b);
write_i2c(0x581e ,0x2a);
write_i2c(0x581f ,0x16);
write_i2c(0x5820 ,0x11);
write_i2c(0x5821 ,0x11);
write_i2c(0x5822 ,0x15);
write_i2c(0x5823 ,0x29);
write_i2c(0x5824 ,0xbf);
write_i2c(0x5825 ,0xaf);
write_i2c(0x5826 ,0x9f);
write_i2c(0x5827 ,0xaf);
write_i2c(0x5828 ,0xdf);
write_i2c(0x5829 ,0x6f);
write_i2c(0x582a ,0x8e);
write_i2c(0x582b ,0xab);
write_i2c(0x582c ,0x9e);
write_i2c(0x582d ,0x7f);
write_i2c(0x582e ,0x4f);
write_i2c(0x582f ,0x89);
write_i2c(0x5830 ,0x86);
write_i2c(0x5831 ,0x98);
write_i2c(0x5832 ,0x6f);
write_i2c(0x5833 ,0x4f);
write_i2c(0x5834 ,0x6e);
write_i2c(0x5835 ,0x7b);
write_i2c(0x5836 ,0x7e);
write_i2c(0x5837 ,0x6f);
write_i2c(0x5838 ,0xde);
write_i2c(0x5839 ,0xbf);
write_i2c(0x583a ,0x9f);

write_i2c(0x583b ,0xbf);
write_i2c(0x583c ,0xec);
write_i2c(0x5025 ,0x00);
write_i2c(0x3a0f ,0x30);
write_i2c(0x3a10 ,0x28);
write_i2c(0x3a1b ,0x30);
write_i2c(0x3a1e ,0x26);
write_i2c(0x3a11 ,0x60);
write_i2c(0x3a1f ,0x14);
write_i2c(0x3a18 ,0x00);
write_i2c(0x3a19 ,0xf8);

13.1.4 QSXGA Capture

13.1.4.1 YUV mode capture

write_i2c(0x3820 ,0x40);
write_i2c(0x3821 ,0x06);
write_i2c(0x3814 ,0x11);
write_i2c(0x3815 ,0x11);
write_i2c(0x3803 ,0x00);
write_i2c(0x3807 ,0x9f);
write_i2c(0x3808 ,0x0a);
write_i2c(0x3809 ,0x20);
write_i2c(0x380a ,0x07);
write_i2c(0x380b ,0x98);
write_i2c(0x380c ,0x0b);
write_i2c(0x380d ,0x1c);
write_i2c(0x380e ,0x07);
write_i2c(0x380f ,0xb0);
write_i2c(0x3813 ,0x04);
write_i2c(0x3618 ,0x04);
write_i2c(0x3612 ,0x4b);
write_i2c(0x3708 ,0x21);
write_i2c(0x3709 ,0x12);
write_i2c(0x370c ,0x00);
write_i2c(0x3a02 ,0x07);
write_i2c(0x3a03 ,0xb0);
write_i2c(0x3a0e ,0x06);
write_i2c(0x3a0d ,0x08);
write_i2c(0x3a14 ,0x07);
write_i2c(0x3a15 ,0xb0);

write_i2c(0x4004 ,0x06);
write_i2c(0x5000 ,0x07);
write_i2c(0x5181 ,0x52);
write_i2c(0x5182 ,0x00);
write_i2c(0x5197 ,0x01);
write_i2c(0x519e ,0x38);
write_i2c(0x3035 ,0x21);
write_i2c(0x5000 ,0x27);
write_i2c(0x5001 ,0x83);
write_i2c(0x3035 ,0x71);
write_i2c(0x4713 ,0x02);
write_i2c(0x3036 ,0x69);
write_i2c(0x4407 ,0x0c);
write_i2c(0x460b ,0x37);
write_i2c(0x460c ,0x20);
write_i2c(0x3824 ,0x01);

13.1.4.2 Jpg mode capture

write_i2c(0x3820 ,0x40);
write_i2c(0x3821 ,0x26);
write_i2c(0x3814 ,0x11);
write_i2c(0x3815 ,0x11);
write_i2c(0x3803 ,0x00);
write_i2c(0x3807 ,0x9f);
write_i2c(0x3808 ,0x0a);
write_i2c(0x3809 ,0x20);
write_i2c(0x380a ,0x07);
write_i2c(0x380b ,0x98);
write_i2c(0x380c ,0x0b);
write_i2c(0x380d ,0x1c);
write_i2c(0x380e ,0x07);
write_i2c(0x380f ,0xb0);
write_i2c(0x3813 ,0x04);
write_i2c(0x3618 ,0x04);
write_i2c(0x3612 ,0x4b);
write_i2c(0x3708 ,0x21);
write_i2c(0x3709 ,0x12);
write_i2c(0x370c ,0x00);

write_i2c(0x3a02 ,0x07);
write_i2c(0x3a03 ,0xb0);
write_i2c(0x3a0e ,0x06);
write_i2c(0x3a0d ,0x08);
write_i2c(0x3a14 ,0x07);
write_i2c(0x3a15 ,0xb0);
write_i2c(0x4001 ,0x02);
write_i2c(0x4004 ,0x06);
write_i2c(0x3002 ,0x00);
write_i2c(0x3006 ,0xff);
write_i2c(0x3824 ,0x04);
write_i2c(0x5001 ,0x83);
write_i2c(0x3036 ,0x69);
write_i2c(0x3035 ,0x31);

13.1.5 Other Capture size DCW from QSXGA

QSXGA to vga(640*480)

write_i2c(0x3800 ,0x1),
write_i2c(0x3801 ,0x8A),
write_i2c(0x3802 ,0x0),
write_i2c(0x3803 ,0xA),
write_i2c(0x3804 ,0xA),
write_i2c(0x3805 ,0x20),
write_i2c(0x3806 ,0x7),
write_i2c(0x3807 ,0x98),
write_i2c(0x3808 ,0x2),
write_i2c(0x3809 ,0x80),
write_i2c(0x380a ,0x1),
write_i2c(0x380b ,0xe0),
write_i2c(0x380c ,0xc),
write_i2c(0x380d ,0x80),
write_i2c(0x380e ,0x7),
write_i2c(0x380f ,0xd0),
write_i2c(0x5001 ,0x7f),
write_i2c(0x5680 ,0x0),

write_i2c(0x5681 ,0x0),
write_i2c(0x5682 ,0xA),
write_i2c(0x5683 ,0x20),
write_i2c(0x5684 ,0x0),
write_i2c(0x5685 ,0x0),
write_i2c(0x5686 ,0x7),
write_i2c(0x5687 ,0x98),

QSXGA to sxga(1280*960)

write_i2c(0x3800 ,0x1),
write_i2c(0x3801 ,0x8A),
write_i2c(0x3802 ,0x0),
write_i2c(0x3803 ,0xA),
write_i2c(0x3804 ,0xA),
write_i2c(0x3805 ,0x20),
write_i2c(0x3806 ,0x7),
write_i2c(0x3807 ,0x98),
write_i2c(0x3808 ,0x5),
write_i2c(0x3809 ,0x0),
write_i2c(0x380a ,0x3),
write_i2c(0x380b ,0xc0),
write_i2c(0x380c ,0xc),
write_i2c(0x380d ,0x80),
write_i2c(0x380e ,0x7),
write_i2c(0x380f ,0xd0),
write_i2c(0x5001 ,0x7f),
write_i2c(0x5680 ,0x0),
write_i2c(0x5681 ,0x0),
write_i2c(0x5682 ,0xA),
write_i2c(0x5683 ,0x20),
write_i2c(0x5684 ,0x0),
write_i2c(0x5685 ,0x0),
write_i2c(0x5686 ,0x7),
write_i2c(0x5687 ,0x98),

QSXGA to QVGA(320*240)

write_i2c(0x3800 ,0x1),
write_i2c(0x3801 ,0x8A),
write_i2c(0x3802 ,0x0),
write_i2c(0x3803 ,0xA),
write_i2c(0x3804 ,0xA),
write_i2c(0x3805 ,0x20),
write_i2c(0x3806 ,0x7),
write_i2c(0x3807 ,0x98),
write_i2c(0x3808 ,0x1),

```
write_i2c(0x3809 ,0x40),
write_i2c(0x380a ,0x0 ),
write_i2c(0x380b ,0xf0),
write_i2c(0x380c ,0xc ),
write_i2c(0x380d ,0x80),
write_i2c(0x380e ,0x7 ),
write_i2c(0x380f ,0xd0),
write_i2c(0x5001 ,0x7f),
write_i2c(0x5680 ,0x0 ),
write_i2c(0x5681 ,0x0 ),
write_i2c(0x5682 ,0xA ),
write_i2c(0x5683 ,0x20),
write_i2c(0x5684 ,0x0 ),
write_i2c(0x5685 ,0x0 ),
write_i2c(0x5686 ,0x7 ),
write_i2c(0x5687 ,0x98),
```

QSXGA to qxga(2048*1536)

```
write_i2c(0x3800 ,0x1 ),
write_i2c(0x3801 ,0x8A),
write_i2c(0x3802 ,0x0 ),
write_i2c(0x3803 ,0xA ),
write_i2c(0x3804 ,0xA ),
write_i2c(0x3805 ,0x20),
write_i2c(0x3806 ,0x7 ),
write_i2c(0x3807 ,0x98),
write_i2c(0x3808 ,0x8 ),
write_i2c(0x3809 ,0x0 ),
write_i2c(0x380a ,0x6 ),
write_i2c(0x380b ,0x0 ),
write_i2c(0x380c ,0xc ),
write_i2c(0x380d ,0x80),
write_i2c(0x380e ,0x7 ),
write_i2c(0x380f ,0xd0),
write_i2c(0x5001 ,0x7f),
write_i2c(0x5680 ,0x0 ),
write_i2c(0x5681 ,0x0 ),
write_i2c(0x5682 ,0xA ),
write_i2c(0x5683 ,0x20),
write_i2c(0x5684 ,0x0 ),
write_i2c(0x5685 ,0x0 ),
write_i2c(0x5686 ,0x7 ),
write_i2c(0x5687 ,0x98),
```

QSXGA to uxga(1600*1200)

```
write_i2c(0x3800 ,0x1 ),
write_i2c(0x3801 ,0x8A),
write_i2c(0x3802 ,0x0 ),
write_i2c(0x3803 ,0xA ),
write_i2c(0x3804 ,0xA ),
write_i2c(0x3805 ,0x20),
write_i2c(0x3806 ,0x7 ),
write_i2c(0x3807 ,0x98),
write_i2c(0x3808 ,0x6 ),
write_i2c(0x3809 ,0x40),
write_i2c(0x380a ,0x4 ),
write_i2c(0x380b ,0xb0),
write_i2c(0x380c ,0xc ),
write_i2c(0x380d ,0x80),
write_i2c(0x380e ,0x7 ),
write_i2c(0x380f ,0xd0),
write_i2c(0x5001 ,0x7f),
write_i2c(0x5680 ,0x0 ),
write_i2c(0x5681 ,0x0 ),
write_i2c(0x5682 ,0xA ),
write_i2c(0x5683 ,0x20),
write_i2c(0x5684 ,0x0 ),
write_i2c(0x5685 ,0x0 ),
write_i2c(0x5686 ,0x7 ),
write_i2c(0x5687 ,0x98),
```

QSXGA to xga(1024*768)

```
write_i2c(0x3800 ,0x1 ),
write_i2c(0x3801 ,0x8A),
write_i2c(0x3802 ,0x0 ),
write_i2c(0x3803 ,0xA ),
write_i2c(0x3804 ,0xA ),
write_i2c(0x3805 ,0x20),
write_i2c(0x3806 ,0x7 ),
write_i2c(0x3807 ,0x98),
write_i2c(0x3808 ,0x4 ),
write_i2c(0x3809 ,0x0 ),
write_i2c(0x380a ,0x3 ),
write_i2c(0x380b ,0x0 ),
write_i2c(0x380c ,0xc ),
write_i2c(0x380d ,0x80),
write_i2c(0x380e ,0x7 ),
write_i2c(0x380f ,0xd0),
write_i2c(0x5001 ,0x7f),
```

write_i2c(0x5680 ,0x0),
write_i2c(0x5681 ,0x0),
write_i2c(0x5682 ,0xA),
write_i2c(0x5683 ,0x20),
write_i2c(0x5684 ,0x0),
write_i2c(0x5685 ,0x0),
write_i2c(0x5686 ,0x7),
write_i2c(0x5687 ,0x98),

13.1.6 Return to yuv mode

13.1.6.1 From 5M YUV to vga YUV

write_i2c(0x3820 ,0x41);
write_i2c(0x3821 ,0x7);
write_i2c(0x3814 ,0x31);
write_i2c(0x3815 ,0x31);
write_i2c(0x3803 ,0x4);
write_i2c(0x3807 ,0x9b);
write_i2c(0x3808 ,0x2);
write_i2c(0x3809 ,0x80);
write_i2c(0x380a ,0x1);
write_i2c(0x380b ,0xe0);
write_i2c(0x380c ,0x7);
write_i2c(0x380d ,0x68);
write_i2c(0x380e ,0x3);
write_i2c(0x380f ,0xd8);
write_i2c(0x3813 ,0x6);
write_i2c(0x3618 ,0x0);
write_i2c(0x3612 ,0x49);
write_i2c(0x3708 ,0x62);
write_i2c(0x3709 ,0x52);
write_i2c(0x370c ,0x3);
write_i2c(0x3a02 ,0x3);
write_i2c(0x3a03 ,0xd8);
write_i2c(0x3a0e ,0x3);
write_i2c(0x3a0d ,0x4);

write_i2c(0x3a14 ,0x3);
write_i2c(0x3a15 ,0xd8);
write_i2c(0x4004 ,0x2);
write_i2c(0x5000 ,0xa7);
write_i2c(0x5181 ,0xf2);
write_i2c(0x5182 ,0x0);
write_i2c(0x5197 ,0x1);
write_i2c(0x519e ,0x38);
write_i2c(0x3035 ,0x21);
write_i2c(0x5000 ,0xa7);
write_i2c(0x5001 ,0xa3);
write_i2c(0x3035 ,0x21);
write_i2c(0x4713 ,0x3);
write_i2c(0x3036 ,0x46);
write_i2c(0x4407 ,0x4);
write_i2c(0x460b ,0x35);
write_i2c(0x460c ,0x22);
write_i2c(0x3824 ,0x2);

13.1.6.2 From 5M Jpg to vga yuv

write_i2c(0x3820 ,0x41);
write_i2c(0x3821 ,0x07);
write_i2c(0x3814 ,0x31);
write_i2c(0x3815 ,0x31);
write_i2c(0x3803 ,0x04);
write_i2c(0x3807 ,0x9b);
write_i2c(0x3808 ,0x02);
write_i2c(0x3809 ,0x80);
write_i2c(0x380a ,0x01);
write_i2c(0x380b ,0xe0);
write_i2c(0x380c ,0x07);
write_i2c(0x380d ,0x68);

write_i2c(0x380e ,0x03);
write_i2c(0x380f ,0xd8);
write_i2c(0x3813 ,0x06);
write_i2c(0x3618 ,0x00);
write_i2c(0x3612 ,0x49);
write_i2c(0x3708 ,0x62);
write_i2c(0x3709 ,0x52);
write_i2c(0x370c ,0x03);
write_i2c(0x3a02 ,0x03);
write_i2c(0x3a03 ,0xd8);
write_i2c(0x3a0e ,0x03);
write_i2c(0x3a0d ,0x04);
write_i2c(0x3a14 ,0x03);
write_i2c(0x3a15 ,0xd8);
write_i2c(0x4001 ,0x02);
write_i2c(0x4004 ,0x02);
write_i2c(0x3002 ,0x1c);
write_i2c(0x3006 ,0xc3);
write_i2c(0x3036 ,0x46);
write_i2c(0x3824 ,0x02);
write_i2c(0x5001 ,0xa3);
write_i2c(0x3035 ,0x21);

13.1.7 YUV and JPEG mode change setting

13.1.7.1 YUV to JPEG setting

write_i2c(0x3002 ,0x00);
write_i2c(0x3006 ,0xff);
write_i2c(0x3821 ,0x27);
write_i2c(0x4300 ,0x30);
write_i2c(0x501f ,0x00);
write_i2c(0x4713 ,0x02);
write_i2c(0x460c ,0x22);
write_i2c(0x3824 ,0x04);
write_i2c(0x460b ,0x35);

13.1.7.2 JPEG to YUV setting

```
write_i2c(0x3002 ,0x1c);
write_i2c(0x3006 ,0xc3);
write_i2c(0x3821 ,0x07);
write_i2c(0x4300 ,0x30);
write_i2c(0x501f ,0x00);
write_i2c(0x460c ,0x20);
write_i2c(0x3824 ,0x04);
write_i2c(0x460b ,0x37);
```

13.2 Sensor Raw setting

```
write_i2c(0x3008 ,0x82);
write_i2c(0x3103 ,0x03);
write_i2c(0x3017 ,0xff);
write_i2c(0x3018 ,0xff);
write_i2c(0x3108 ,0x01);
write_i2c(0x3037 ,0x13);
write_i2c(0x3630 ,0x2e);
write_i2c(0x3632 ,0xe2);
write_i2c(0x3633 ,0x23);
write_i2c(0x3634 ,0x44);
write_i2c(0x3621 ,0xe0);
write_i2c(0x3704 ,0xa0);
write_i2c(0x3703 ,0x5a);
write_i2c(0x3715 ,0x78);
write_i2c(0x3717 ,0x01);
write_i2c(0x370b ,0x60);
write_i2c(0x3705 ,0x1a);
write_i2c(0x3905 ,0x02);
write_i2c(0x3906 ,0x10);
write_i2c(0x3901 ,0x0a);
write_i2c(0x3731 ,0x12);
write_i2c(0x3600 ,0x08);
write_i2c(0x3601 ,0x33);
write_i2c(0x471c ,0x50);
write_i2c(0x3820 ,0x40);
write_i2c(0x3821 ,0x00);
write_i2c(0x3814 ,0x11);
write_i2c(0x3815 ,0x11);
write_i2c(0x3800 ,0x00);
write_i2c(0x3801 ,0x00);
write_i2c(0x3802 ,0x00);
write_i2c(0x3803 ,0x00);
write_i2c(0x3804 ,0x0a);
write_i2c(0x3805 ,0x3f);
```

```
write_i2c(0x3806 ,0x07);
write_i2c(0x3807 ,0x9f);
write_i2c(0x3808 ,0x0a);
write_i2c(0x3809 ,0x20);
write_i2c(0x380a ,0x07);
write_i2c(0x380b ,0x98);
write_i2c(0x380c ,0x0b);
write_i2c(0x380d ,0x1c);
write_i2c(0x380e ,0x07);
write_i2c(0x380f ,0xb0);
write_i2c(0x3810 ,0x00);
write_i2c(0x3811 ,0x10);
write_i2c(0x3812 ,0x00);
write_i2c(0x3813 ,0x04);
write_i2c(0x3618 ,0x04);
write_i2c(0x3612 ,0x4b);
write_i2c(0x3708 ,0x21);
write_i2c(0x3709 ,0x12);
write_i2c(0x370c ,0x00);
write_i2c(0x3a02 ,0x07);
write_i2c(0x3a03 ,0xb0);
write_i2c(0x3a08 ,0x01);
write_i2c(0x3a09 ,0x27);
write_i2c(0x3a0a ,0x00);
write_i2c(0x3a0b ,0xf6);
write_i2c(0x3a0e ,0x06);
write_i2c(0x3a0d ,0x08);
write_i2c(0x3a14 ,0x07);
write_i2c(0x3a15 ,0xb0);
write_i2c(0x4001 ,0x02);
write_i2c(0x4004 ,0x06);
write_i2c(0x3000 ,0x00);
write_i2c(0x3002 ,0x1c);
write_i2c(0x3004 ,0xff);
write_i2c(0x3006 ,0xc3);
write_i2c(0x4300 ,0xf8);
write_i2c(0x5001 ,0x00);
write_i2c(0x501f ,0x03);
write_i2c(0x5000 ,0x06);
write_i2c(0x3a0f ,0x36);
write_i2c(0x3a10 ,0x2e);
write_i2c(0x3a1b ,0x38);
write_i2c(0x3a1e ,0x2c);
write_i2c(0x3a11 ,0x70);
write_i2c(0x3a1f ,0x18);
write_i2c(0x3a18 ,0x00);
write_i2c(0x3a19 ,0xf8);
write_i2c(0x3035 ,0x41);
```

13. 3 High Resolution Video

13.3.1 1080 P

```
write_i2c(0x3008 ,0x82);
write_i2c(0x3103 ,0x03);
write_i2c(0x3017 ,0xff);
write_i2c(0x3018 ,0xff);
write_i2c(0x3108 ,0x01);
write_i2c(0x3037 ,0x13);
write_i2c(0x3630 ,0x2e);
write_i2c(0x3632 ,0xe2);
write_i2c(0x3633 ,0x23);
write_i2c(0x3634 ,0x44);
write_i2c(0x3621 ,0xe0);
write_i2c(0x3704 ,0xa0);
write_i2c(0x3703 ,0x5a);
write_i2c(0x3715 ,0x78);
write_i2c(0x3717 ,0x01);
write_i2c(0x370b ,0x60);
write_i2c(0x3705 ,0x1a);
write_i2c(0x3905 ,0x02);
write_i2c(0x3906 ,0x10);
write_i2c(0x3901 ,0x0a);
write_i2c(0x3731 ,0x12);
write_i2c(0x3600 ,0x08);
write_i2c(0x3601 ,0x33);
write_i2c(0x471c ,0x50);
write_i2c(0x3820 ,0x41);
write_i2c(0x3821 ,0x26);
write_i2c(0x3814 ,0x11);
write_i2c(0x3815 ,0x11);
write_i2c(0x3800 ,0x01);
write_i2c(0x3801 ,0x50);
write_i2c(0x3802 ,0x01);
write_i2c(0x3803 ,0xb2);
write_i2c(0x3804 ,0x08);
write_i2c(0x3805 ,0xef);
write_i2c(0x3806 ,0x05);
write_i2c(0x3807 ,0xf2);
write_i2c(0x3808 ,0x07);
write_i2c(0x3809 ,0x80);
write_i2c(0x380a ,0x04);
write_i2c(0x380b ,0x38);
write_i2c(0x380c ,0x09);
write_i2c(0x380d ,0xc4);
write_i2c(0x380e ,0x04);
write_i2c(0x380f ,0x60);
```

```
write_i2c(0x3810 ,0x00);
write_i2c(0x3811 ,0x10);
write_i2c(0x3812 ,0x00);
write_i2c(0x3813 ,0x04);
write_i2c(0x3618 ,0x04);
write_i2c(0x3612 ,0x4b);
write_i2c(0x3708 ,0x62);
write_i2c(0x3709 ,0x12);
write_i2c(0x370c ,0x00);
write_i2c(0x3a02 ,0x04);
write_i2c(0x3a03 ,0x60);
write_i2c(0x3a08 ,0x01);
write_i2c(0x3a09 ,0x50);
write_i2c(0x3a0a ,0x01);
write_i2c(0x3a0b ,0x18);
write_i2c(0x3a0e ,0x03);
write_i2c(0x3a0d ,0x04);
write_i2c(0x3a14 ,0x04);
write_i2c(0x3a15 ,0x60);
write_i2c(0x4001 ,0x02);
write_i2c(0x4004 ,0x06);
write_i2c(0x3002 ,0x00);
write_i2c(0x3006 ,0xff);
write_i2c(0x4300 ,0x30);
write_i2c(0x501f ,0x00);
write_i2c(0x4713 ,0x03);
write_i2c(0x3035 ,0x11);
write_i2c(0x4407 ,0x04);
write_i2c(0x460b ,0x35);
write_i2c(0x460c ,0x22);
write_i2c(0x3824 ,0x04);
write_i2c(0x5000 ,0xa7);
write_i2c(0x5001 ,0x83);
write_i2c(0x5000 ,0xa7);
write_i2c(0x3622 ,0x01);
write_i2c(0x3635 ,0x1c);
write_i2c(0x3634 ,0x40);
write_i2c(0x3c01 ,0x34);
write_i2c(0x3c00 ,0x00);
write_i2c(0x3c04 ,0x28);
write_i2c(0x3c05 ,0x98);
write_i2c(0x3c06 ,0x00);
write_i2c(0x3c07 ,0x08);
write_i2c(0x3c08 ,0x00);
write_i2c(0x3c09 ,0x1c);
write_i2c(0x300c ,0x22);
write_i2c(0x3c0a ,0x9c);
write_i2c(0x3c0b ,0x40);
```

```
write_i2c(0x5180 ,0xff);
write_i2c(0x5181 ,0xf2);
write_i2c(0x5182 ,0x00);
write_i2c(0x5183 ,0x94);
write_i2c(0x5184 ,0x25);
write_i2c(0x5185 ,0x24);
write_i2c(0x5186 ,0x06);
write_i2c(0x5187 ,0x08);
write_i2c(0x5188 ,0x08);
write_i2c(0x5189 ,0x78);
write_i2c(0x518a ,0x54);
write_i2c(0x518b ,0xb2);
write_i2c(0x518c ,0xb2);
write_i2c(0x518d ,0x44);
write_i2c(0x518e ,0x3d);
write_i2c(0x518f ,0x58);
write_i2c(0x5190 ,0x46);
write_i2c(0x5191 ,0xf8);
write_i2c(0x5192 ,0x04);
write_i2c(0x5193 ,0x70);
write_i2c(0x5194 ,0xf0);
write_i2c(0x5195 ,0xf0);
write_i2c(0x5196 ,0x03);
write_i2c(0x5197 ,0x01);
write_i2c(0x5198 ,0x04);
write_i2c(0x5199 ,0x12);
write_i2c(0x519a ,0x04);
write_i2c(0x519b ,0x00);
write_i2c(0x519c ,0x06);
write_i2c(0x519d ,0x82);
write_i2c(0x519e ,0x38);
write_i2c(0x5381 ,0x1c);
write_i2c(0x5382 ,0x5a);
write_i2c(0x5383 ,0x06);
write_i2c(0x5384 ,0x20);
write_i2c(0x5385 ,0x80);
write_i2c(0x5386 ,0xa0);
write_i2c(0x5387 ,0xa2);
write_i2c(0x5388 ,0xa0);
write_i2c(0x5389 ,0x02);
write_i2c(0x538a ,0x01);
write_i2c(0x538b ,0x98);
write_i2c(0x5300 ,0x08);
write_i2c(0x5301 ,0x30);
write_i2c(0x5302 ,0x10);
write_i2c(0x5303 ,0x00);
write_i2c(0x5304 ,0x08);
write_i2c(0x5305 ,0x30);
```

```
write_i2c(0x5306 ,0x08);
write_i2c(0x5307 ,0x16);
write_i2c(0x5309 ,0x08);
write_i2c(0x530a ,0x30);
write_i2c(0x530b ,0x04);
write_i2c(0x530c ,0x06);
write_i2c(0x5480 ,0x01);
write_i2c(0x5481 ,0x08);
write_i2c(0x5482 ,0x14);
write_i2c(0x5483 ,0x28);
write_i2c(0x5484 ,0x51);
write_i2c(0x5485 ,0x65);
write_i2c(0x5486 ,0x71);
write_i2c(0x5487 ,0x7d);
write_i2c(0x5488 ,0x87);
write_i2c(0x5489 ,0x91);
write_i2c(0x548a ,0x9a);
write_i2c(0x548b ,0xaa);
write_i2c(0x548c ,0xb8);
write_i2c(0x548d ,0xcd);
write_i2c(0x548e ,0xdd);
write_i2c(0x548f ,0xea);
write_i2c(0x5490 ,0x1d);
write_i2c(0x5580 ,0x02);
write_i2c(0x5583 ,0x40);
write_i2c(0x5584 ,0x10);
write_i2c(0x5589 ,0x10);
write_i2c(0x558a ,0x00);
write_i2c(0x558b ,0xf8);
write_i2c(0x5800 ,0x23);
write_i2c(0x5801 ,0x15);
write_i2c(0x5802 ,0x10);
write_i2c(0x5803 ,0x10);
write_i2c(0x5804 ,0x15);
write_i2c(0x5805 ,0x23);
write_i2c(0x5806 ,0x0c);
write_i2c(0x5807 ,0x08);
write_i2c(0x5808 ,0x05);
write_i2c(0x5809 ,0x05);
write_i2c(0x580a ,0x08);
write_i2c(0x580b ,0x0c);
write_i2c(0x580c ,0x07);
write_i2c(0x580d ,0x03);
write_i2c(0x580e ,0x00);
write_i2c(0x580f ,0x00);
write_i2c(0x5810 ,0x03);
write_i2c(0x5811 ,0x07);
write_i2c(0x5812 ,0x07);
```

```
write_i2c(0x5813 ,0x03);
write_i2c(0x5814 ,0x00);
write_i2c(0x5815 ,0x00);
write_i2c(0x5816 ,0x03);
write_i2c(0x5817 ,0x07);
write_i2c(0x5818 ,0x0b);
write_i2c(0x5819 ,0x08);
write_i2c(0x581a ,0x05);
write_i2c(0x581b ,0x05);
write_i2c(0x581c ,0x07);
write_i2c(0x581d ,0x0b);
write_i2c(0x581e ,0x2a);
write_i2c(0x581f ,0x16);
write_i2c(0x5820 ,0x11);
write_i2c(0x5821 ,0x11);
write_i2c(0x5822 ,0x15);
write_i2c(0x5823 ,0x29);
write_i2c(0x5824 ,0xbf);
write_i2c(0x5825 ,0xaf);
write_i2c(0x5826 ,0x9f);
write_i2c(0x5827 ,0xaf);
write_i2c(0x5828 ,0xdf);
write_i2c(0x5829 ,0x6f);
write_i2c(0x582a ,0x8e);
write_i2c(0x582b ,0xab);
write_i2c(0x582c ,0x9e);
write_i2c(0x582d ,0x7f);
write_i2c(0x582e ,0x4f);
write_i2c(0x582f ,0x89);
write_i2c(0x5830 ,0x86);
write_i2c(0x5831 ,0x98);
write_i2c(0x5832 ,0x6f);
write_i2c(0x5833 ,0x4f);
write_i2c(0x5834 ,0x6e);
write_i2c(0x5835 ,0x7b);
write_i2c(0x5836 ,0x7e);
write_i2c(0x5837 ,0x6f);
write_i2c(0x5838 ,0xde);
write_i2c(0x5839 ,0xbf);
write_i2c(0x583a ,0x9f);
write_i2c(0x583b ,0xbf);
write_i2c(0x583c ,0xec);
write_i2c(0x5025 ,0x00);
write_i2c(0x3a0f ,0x30);
write_i2c(0x3a10 ,0x28);
write_i2c(0x3a1b ,0x30);
write_i2c(0x3a1e ,0x26);
write_i2c(0x3a11 ,0x60);
```

```
write_i2c(0x3a1f ,0x14);
write_i2c(0x3a18 ,0x00);
write_i2c(0x3a19 ,0xf8);
write_i2c(0x3035 ,0x21);
```

13.3.2 720 P

```
write_i2c(0x3008 ,0x82);
write_i2c(0x3103 ,0x03);
write_i2c(0x3017 ,0xff);
write_i2c(0x3018 ,0xff);
write_i2c(0x3108 ,0x01);
write_i2c(0x3037 ,0x13);
write_i2c(0x3630 ,0x2e);
write_i2c(0x3632 ,0xe2);
write_i2c(0x3633 ,0x23);
write_i2c(0x3634 ,0x44);
write_i2c(0x3621 ,0xe0);
write_i2c(0x3704 ,0xa0);
write_i2c(0x3703 ,0x5a);
write_i2c(0x3715 ,0x78);
write_i2c(0x3717 ,0x01);
write_i2c(0x370b ,0x60);
write_i2c(0x3705 ,0x1a);
write_i2c(0x3905 ,0x02);
write_i2c(0x3906 ,0x10);
write_i2c(0x3901 ,0x0a);
write_i2c(0x3731 ,0x12);
write_i2c(0x3600 ,0x08);
write_i2c(0x3601 ,0x33);
write_i2c(0x471c ,0x50);
write_i2c(0x3820 ,0x41);
write_i2c(0x3821 ,0x27);
write_i2c(0x3814 ,0x31);
write_i2c(0x3815 ,0x31);
write_i2c(0x3800 ,0x00);
write_i2c(0x3801 ,0x00);
write_i2c(0x3802 ,0x00);
write_i2c(0x3803 ,0xfa);
write_i2c(0x3804 ,0x0a);
write_i2c(0x3805 ,0x3f);
write_i2c(0x3806 ,0x06);
write_i2c(0x3807 ,0xa9);
write_i2c(0x3808 ,0x05);
write_i2c(0x3809 ,0x00);
write_i2c(0x380a ,0x02);
write_i2c(0x380b ,0xd0);
write_i2c(0x380c ,0x07);
write_i2c(0x380d ,0x64);
```

```
write_i2c(0x380e ,0x02);
write_i2c(0x380f ,0xe4);
write_i2c(0x3810 ,0x00);
write_i2c(0x3811 ,0x10);
write_i2c(0x3812 ,0x00);
write_i2c(0x3813 ,0x04);
write_i2c(0x3618 ,0x00);
write_i2c(0x3612 ,0x49);
write_i2c(0x3708 ,0x62);
write_i2c(0x3709 ,0x52);
write_i2c(0x370c ,0x03);
write_i2c(0x3a02 ,0x02);
write_i2c(0x3a03 ,0xe4);
write_i2c(0x3a08 ,0x01);
write_i2c(0x3a09 ,0xbc);
write_i2c(0x3a0a ,0x01);
write_i2c(0x3a0b ,0x72);
write_i2c(0x3a0e ,0x01);
write_i2c(0x3a0d ,0x02);
write_i2c(0x3a14 ,0x02);
write_i2c(0x3a15 ,0xe4);
write_i2c(0x4001 ,0x02);
write_i2c(0x4004 ,0x02);
write_i2c(0x3002 ,0x00);
write_i2c(0x3006 ,0xff);
write_i2c(0x4300 ,0x30);
write_i2c(0x501f ,0x00);
write_i2c(0x4713 ,0x03);
write_i2c(0x3035 ,0x11);
write_i2c(0x4407 ,0x04);
write_i2c(0x460b ,0x35);
write_i2c(0x460c ,0x22);
write_i2c(0x3824 ,0x04);
write_i2c(0x5000 ,0xa7);
write_i2c(0x5001 ,0x83);
write_i2c(0x5000 ,0xa7);
write_i2c(0x3622 ,0x01);
write_i2c(0x3635 ,0x1c);
write_i2c(0x3634 ,0x40);
write_i2c(0x3c01 ,0x34);
write_i2c(0x3c00 ,0x00);
write_i2c(0x3c04 ,0x28);
write_i2c(0x3c05 ,0x98);
write_i2c(0x3c06 ,0x00);
write_i2c(0x3c07 ,0x08);
write_i2c(0x3c08 ,0x00);
write_i2c(0x3c09 ,0x1c);
write_i2c(0x300c ,0x22);
```

```
write_i2c(0x3c0a ,0x9c);
write_i2c(0x3c0b ,0x40);
write_i2c(0x5180 ,0xff);
write_i2c(0x5181 ,0xf2);
write_i2c(0x5182 ,0x00);
write_i2c(0x5183 ,0x94);
write_i2c(0x5184 ,0x25);
write_i2c(0x5185 ,0x24);
write_i2c(0x5186 ,0x06);
write_i2c(0x5187 ,0x08);
write_i2c(0x5188 ,0x08);
write_i2c(0x5189 ,0x78);
write_i2c(0x518a ,0x54);
write_i2c(0x518b ,0xb2);
write_i2c(0x518c ,0xb2);
write_i2c(0x518d ,0x44);
write_i2c(0x518e ,0x3d);
write_i2c(0x518f ,0x58);
write_i2c(0x5190 ,0x46);
write_i2c(0x5191 ,0xf8);
write_i2c(0x5192 ,0x04);
write_i2c(0x5193 ,0x70);
write_i2c(0x5194 ,0xf0);
write_i2c(0x5195 ,0xf0);
write_i2c(0x5196 ,0x03);
write_i2c(0x5197 ,0x01);
write_i2c(0x5198 ,0x04);
write_i2c(0x5199 ,0x12);
write_i2c(0x519a ,0x04);
write_i2c(0x519b ,0x00);
write_i2c(0x519c ,0x06);
write_i2c(0x519d ,0x82);
write_i2c(0x519e ,0x38);
write_i2c(0x5381 ,0x1c);
write_i2c(0x5382 ,0x5a);
write_i2c(0x5383 ,0x06);
write_i2c(0x5384 ,0x20);
write_i2c(0x5385 ,0x80);
write_i2c(0x5386 ,0xa0);
write_i2c(0x5387 ,0xa2);
write_i2c(0x5388 ,0xa0);
write_i2c(0x5389 ,0x02);
write_i2c(0x538a ,0x01);
write_i2c(0x538b ,0x98);
write_i2c(0x5300 ,0x08);
write_i2c(0x5301 ,0x30);
write_i2c(0x5302 ,0x10);
write_i2c(0x5303 ,0x00);
```

```
write_i2c(0x5304 ,0x08);
write_i2c(0x5305 ,0x30);
write_i2c(0x5306 ,0x08);
write_i2c(0x5307 ,0x16);
write_i2c(0x5309 ,0x08);
write_i2c(0x530a ,0x30);
write_i2c(0x530b ,0x04);
write_i2c(0x530c ,0x06);
write_i2c(0x5480 ,0x01);
write_i2c(0x5481 ,0x08);
write_i2c(0x5482 ,0x14);
write_i2c(0x5483 ,0x28);
write_i2c(0x5484 ,0x51);
write_i2c(0x5485 ,0x65);
write_i2c(0x5486 ,0x71);
write_i2c(0x5487 ,0x7d);
write_i2c(0x5488 ,0x87);
write_i2c(0x5489 ,0x91);
write_i2c(0x548a ,0x9a);
write_i2c(0x548b ,0xaa);
write_i2c(0x548c ,0xb8);
write_i2c(0x548d ,0xcd);
write_i2c(0x548e ,0xdd);
write_i2c(0x548f ,0xea);
write_i2c(0x5490 ,0x1d);
write_i2c(0x5580 ,0x02);
write_i2c(0x5583 ,0x40);
write_i2c(0x5584 ,0x10);
write_i2c(0x5589 ,0x10);
write_i2c(0x558a ,0x00);
write_i2c(0x558b ,0xf8);
write_i2c(0x5800 ,0x23);
write_i2c(0x5801 ,0x15);
write_i2c(0x5802 ,0x10);
write_i2c(0x5803 ,0x10);
write_i2c(0x5804 ,0x15);
write_i2c(0x5805 ,0x23);
write_i2c(0x5806 ,0x0c);
write_i2c(0x5807 ,0x08);
write_i2c(0x5808 ,0x05);
write_i2c(0x5809 ,0x05);
write_i2c(0x580a ,0x08);
write_i2c(0x580b ,0x0c);
write_i2c(0x580c ,0x07);
write_i2c(0x580d ,0x03);
write_i2c(0x580e ,0x00);
write_i2c(0x580f ,0x00);
write_i2c(0x5810 ,0x03);
```

```
write_i2c(0x5811 ,0x07);
write_i2c(0x5812 ,0x07);
write_i2c(0x5813 ,0x03);
write_i2c(0x5814 ,0x00);
write_i2c(0x5815 ,0x00);
write_i2c(0x5816 ,0x03);
write_i2c(0x5817 ,0x07);
write_i2c(0x5818 ,0x0b);
write_i2c(0x5819 ,0x08);
write_i2c(0x581a ,0x05);
write_i2c(0x581b ,0x05);
write_i2c(0x581c ,0x07);
write_i2c(0x581d ,0x0b);
write_i2c(0x581e ,0x2a);
write_i2c(0x581f ,0x16);
write_i2c(0x5820 ,0x11);
write_i2c(0x5821 ,0x11);
write_i2c(0x5822 ,0x15);
write_i2c(0x5823 ,0x29);
write_i2c(0x5824 ,0xbf);
write_i2c(0x5825 ,0xaf);
write_i2c(0x5826 ,0x9f);
write_i2c(0x5827 ,0xaf);
write_i2c(0x5828 ,0xdf);
write_i2c(0x5829 ,0x6f);
write_i2c(0x582a ,0x8e);
write_i2c(0x582b ,0xab);
write_i2c(0x582c ,0x9e);
write_i2c(0x582d ,0x7f);
write_i2c(0x582e ,0x4f);
write_i2c(0x582f ,0x89);
write_i2c(0x5830 ,0x86);
write_i2c(0x5831 ,0x98);
write_i2c(0x5832 ,0x6f);
write_i2c(0x5833 ,0x4f);
write_i2c(0x5834 ,0x6e);
write_i2c(0x5835 ,0x7b);
write_i2c(0x5836 ,0x7e);
write_i2c(0x5837 ,0x6f);
write_i2c(0x5838 ,0xde);
write_i2c(0x5839 ,0xbf);
write_i2c(0x583a ,0x9f);
write_i2c(0x583b ,0xbf);
write_i2c(0x583c ,0xec);
write_i2c(0x5025 ,0x00);
write_i2c(0x3a0f ,0x30);
write_i2c(0x3a10 ,0x28);
write_i2c(0x3a1b ,0x30);
```

```
write_i2c(0x3a1e ,0x26);
write_i2c(0x3a11 ,0x60);
write_i2c(0x3a1f ,0x14);
write_i2c(0x3a18 ,0x00);
write_i2c(0x3a19 ,0xf8);
write_i2c(0x3035 ,0x21);
```


14. Capture Sequence

14.1 Shutter

The shutter of OV5640 controls exposure time. The unit of shutter is line period.

Shutter value has limitation for each output resolution. The limitation is stored in 2 registers, reg0x350c, reg0x350d

Maxlines = reg0x350c<<8 + reg0x350d

The shutter value are stored in 3 registers, reg0x3500, reg0x3501,reg3502 .

Shutter = reg0x3500<<12+ reg0x3501<<4+reg3502>>4;

14. 2 Dummy Lines

If enable auto-night mode in ov5640, the maxlines should also be changed

14. 3 Dummy Pixels

For ov5640, use dummy lines to change the fps is recommended, if there are some timing restrict, and dummy pixels must be used, please let OV FAE know.

14.4 Gain

Gain is stored in reg0x350a and Reg0x350b. If only use the gain of Reg0x350b, maximum gain of 32x could be reached. It is enough for camera phone. So we don't discuss reg0x350a here.

Gain = (((reg0x350b & 0xf7)>>4) + 1)*(1 + (reg0x350b & 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.
For 50Hz, the banding filter calculation is

Banding_Filter = Capture_FrameRate * Capture_maxlines /100

For 60Hz, the banding filter calculation is

Banding_Filter = Capture_FrameRate * Capture_maxlines /120

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 OV5640 for preview
Download the setting to initial ov5640

14.7.2 Stop AEC/AGC

write_i2c(0x3503 ,0x07) to stop AGC/AEC.

14.7.3 Single Focus for AF Module

Step 1: Read out state register value state_current,
if(state_current == STATE_INF)goto step2;
else
go to step1;
Step 2: Write cmd_Capture(0x03) to command register(0x3f00);

14.7.4 Read preview register Value

Read back the preview exposure , gain, maxlines.
For preview exposure, read 0x3500,0x3501,0x3502;
For preview gain, read 0x350b;
For preview maxlines, read 0x350c,0x350d;

14.7.5 Change resolution to QSXGA

Download QSXGA setting

14.7.6 Read capture register Value

Read back the capture maxlines. read 0x350c,0x350d

14.7.7 Calculate Capture Exposure from preview

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

14.7.8 Calculate the banding filter value

If use 60Hz

capture banding filter = Capture_Framerate * Capture_MaxLines/120;

if use 50 Hz

capture banding filter = Capture_Framerate * Capture_MaxLines/100;

14.7.9 Redistribute Exposure/Gain with target brightness unchanged

Capture_Exposure_Gain = Capture_Exposure * Gain;

If Capture_Exposure_Gain less than capture maxlines, let Capture exposure =n * capture banding filter; or not, Capture exposure = capture maxlines。

Capture_Gain = Capture_Exposure_Gain/Capture exposure;

14.7.10 write back the gain/exposure value

Write Capture exposure to 0x3500,0x3501,0x3502 and write gain to 0x350b

14.7.11 Capture

Wait for 2 Vsync

Capture the 3rd frame.

14.7.12 Send finish command for AF module

Refer to 16.2.2 for detailed operation.

14.7.13 Back to preview

//Write Registers, Change to VGA

```
//Start AG/AE
write_i2c(0x3503, 0);
```

14.8 Capture reference code

```
        #define  Capture_Framerate 375
        #define   Preview_FrameRate 1500
        BYTE   R0x350b,R0x3502,R0x3501,R0x3500,
        BYTE   Rcap0x350c,Rcap0x350d,Rpre0x350c,Rpre0x350d;
        int      Capture_Framerate;
        int      Lines_10ms;
        int      Capture_MaxLines;
        int      Preview_FrameRate;
        int      Preview_Maxlines;
        long    ulCapture_Exposure;
        long    ulCapture_Exposure_Gain;
        long    ulPreviewExposure;
        long    iCapture_Gain;

//stop AEC/AGC here :write_i2c(0x3503 ,0x07)
//read the registers value to the BYTE 0x350*  parameters.
        BYTE Gain = (BYTE)R0x350b;
        BYTE ExposureLow = (BYTE)R0x3502;
        BYTE ExposureMid = (BYTE)R0x3501;
        BYTE ExposureHigh = (BYTE)R0x3500;
        BYTE PreviewMaxlineHigh = (BYTE)Rpre0x350c;
        BYTE PreviewMaxlineLow = (BYTE)Rpre0x350d;
//change resolution from VGA to QXSGA here
//read the registers value to the BYTE 0x350c and 0x350d  parameters.
        BYTE CaptureMaxlineHigh = (BYTE)Rcap0x350c;
        BYTE CaptureMaxlineLow = (BYTE)Rcap0x350d;

        Preview_Maxlines = 256*PreviewMaxlineHigh + PreviewMaxlineLow;
        Capture_MaxLines = 256*CaptureMaxlineHigh + CaptureMaxlineLow;
        if(m_60Hz== true)
        {
                Lines_10ms = Capture_Framerate * Capture_MaxLines/12000;
        }
        else
        {
                Lines_10ms = Capture_Framerate * Capture_MaxLines/10000;
        }

      ulPreviewExposure  = ((ULONG)(ExposureHigh))<<12 ;
      ulPreviewExposure += ((ULONG)ExposureMid)<<4 ;
      ulPreviewExposure += (ExposureLow >>4);
```

```
  if(0 == Preview_Maxlines ||0 ==Preview_FrameRate ||0== Lines_10ms)
  {
   return;
  }



    ulCapture_Exposure =
          (ulPreviewExposure*(Capture_Framerate)*(Capture_MaxLines))/
(((Preview_Maxlines)*(Preview_FrameRate)));


  iCapture_Gain = (Gain & 0x0f) + 16;



  if (Gain & 0x10)
  {
   iCapture_Gain = iCapture_Gain << 1;
  }
  if (Gain & 0x20)
  {
   iCapture_Gain = iCapture_Gain << 1;
  }
  if (Gain & 0x40)
  {
   iCapture_Gain = iCapture_Gain << 1;
  }
  if (Gain & 0x80)
  {
   iCapture_Gain = iCapture_Gain << 1;
  }

  ulCapture_Exposure_Gain = ulCapture_Exposure * iCapture_Gain;


  if(ulCapture_Exposure_Gain < ((LONG)(Capture_MaxLines)*16))
  {

   ulCapture_Exposure = ulCapture_Exposure_Gain/16;
   if (ulCapture_Exposure > Lines_10ms)
   {
     ulCapture_Exposure /= Lines_10ms;
     ulCapture_Exposure *= Lines_10ms;
   }
  }
  else
  {
```

```
  ulCapture_Exposure = Capture_MaxLines;
 }

 if(ulCapture_Exposure == 0)
 {
    ulCapture_Exposure = 1;
 }

 iCapture_Gain = (ulCapture_Exposure_Gain*2/ulCapture_Exposure + 1)/2;



 ExposureLow =  ((BYTE)ulCapture_Exposure)<<4;
 ExposureMid = (BYTE)(ulCapture_Exposure >> 4) & 0xff;
 ExposureHigh = (BYTE)(ulCapture_Exposure >> 12);


ulCapture_Exposure_end=ulCapture_Exposure;
iCapture_Gain_end=iCapture_Gain;



 Gain = 0;
 if (iCapture_Gain > 31)
 {
  Gain |= 0x10;
  iCapture_Gain = iCapture_Gain >> 1;

 }
 if (iCapture_Gain > 31)
 {
  Gain |= 0x20;
  iCapture_Gain = iCapture_Gain >> 1;
 }
 if (iCapture_Gain > 31)
 {
  Gain |= 0x40;
  iCapture_Gain = iCapture_Gain >> 1;
 }
 if (iCapture_Gain > 31)
 {
  Gain |= 0x80;
  iCapture_Gain = iCapture_Gain >> 1;
 }

 if (iCapture_Gain > 16)
 {
```

```
    Gain |= ((iCapture_Gain -16) & 0x0f);
  }
  if(Gain==0x10)
  {Gain=0x11;}

// write the gain and exposure to 0x350* registers
        m_iWrite0x350b=Gain;
        m_iWrite0x3502=ExposureLow;
        m_iWrite0x3501=ExposureMid;
        m_iWrite0x3500=ExposureHigh;
```

15. Strobe Flash Control

To achieve best image quality in low light condition, strobe flash is recommended.

OV5640 can output one programmable signal from strobe pin.

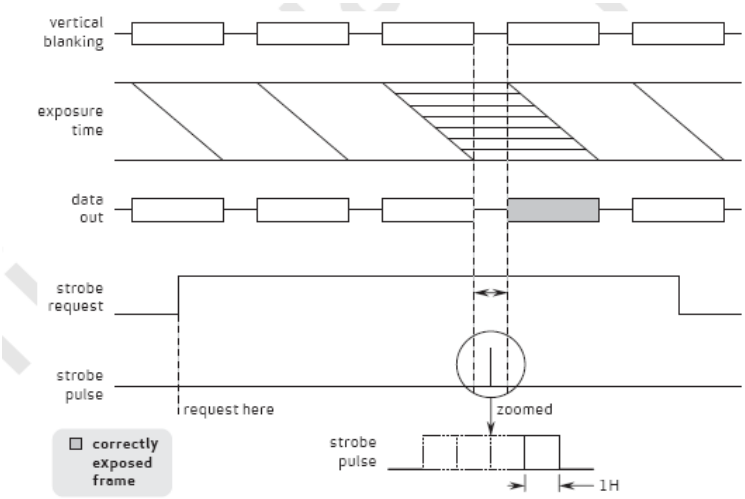
address	register name	default value	R/W	description
0x3B00	STROBE CTRL	0x00	RW	Strobe Control Bit[7]: Strobe request ON/OFF 0: OFF/BLC 1: ON Bit[6]: Strobe pulse reverse Bit[3:2]: width_in_xenon 00: 1H 01: 2H 10: 3H 11: 4H Bit[1:0]: Strobe mode 00: xenon 01: LED 1 10: LED 2 11: LED 3
0x3B04	STROBE FREX EXP	0x04	RW	Strobe FREX Explore High
0x3B05	STROBE FREX EXP	0x00	RW	Strobe FREX Explore Low
0x3B06	FREX CTRL	0x04	RW	FREX Control Bit[7:4]: FREX frame delay number
0x3B07	FREX MODE SEL	0x08	RW	FREX Mode Select Bit[1:0]: FREX mode select 0x: Rolling strobe 10: FREX strobe mode 1 11: FREX strobe mode 2
0x3B08	FREX EXPLORE REQ	0x00	RW	Strobe FREX Explore Request

15.1 Strobe Pulse

The strobe signal is programmable. It support LED and Xenon mode.

15.2 Xenon flash control

After a strobe request is submitted, the strobe pulse will be activated at the beginning of the third frame . The third frame will be correctly exposed. The pulse width can be changed in Xenon mode between 1H and 4H, where H is one horizontal period.



15.3 Application for Xenon flash

Xenon control through STROBE pin of OV5640, STROBE will go high one-four lines period at the VSYNC blanking period when active.

Turn on strobe pin (to high) in the 3rd frame Vsync blanking period

78 3b00 00 80;clear bit[7]

78 3b00 80 80;set bit[7]

Turn off strobe pin

78 3b00 00 80; bit[7],from 1 to 0, turn off strobe pin

15.4 Capture flow with Xenon flash

OV5640 Recommended capturing sequence with using Xenon Flash.

Step 01 Stop frame out

Step 02 Stop AEC/AGC and set AEC value to maximum if it is not maximum

Step 03 Change the resolution for capture mode

Step 04 Resume frame out

Step 05 Issue Xenon flash start

Step 06 Still Image capture at 3rd frame while Xenon flash will be light on

Step 07 Stop frame out and to step 9 if capture is successful, else back to step 6

Step 08 Change back the resolution for preview

Step 09 Start AEC/AGC

Step 10 Resume frame out

16. Auto Focus Application Solution

16.1 Embedded Resources

Embedded Micro-controller

6KB of embedded program memory

Two general purpose IO ports (GPIO0, GPIO1). If the output format of OV5640 is not 10-bit RGB RAW format, Y0, Y1 could also be used as GPIO.

Built-in Auto Focus Control (AFC) functions. AFC module collects edge information for at least five programmable zones

16.2 Embedded Auto Focus Solution

OmniVision had built-in the embedded auto focus control in firmware of OV5640. Currently, the auto focus firmware supports following auto focus camera modules:

1. OV5640 + VCM + AD5820
2. OV5640 + VCM + AD5822
3. OV5640 + VCM + AD5827
4. OV5640 + VCM + DW9710

If you need to support other VCM driver IC, please contact with OmniVision local FAE.

16.3 General Auto Focus Control Flow

Step1. OV5640 Initialization. Refer to 13.1 VGA preview settings.

Step2. Download firmware to built-in memory of OV5640. Confirm VCM type and contact OmniVision local FAE to get related auto focus firmware.

Step3. Send Firmware Commands to OV5640 to control Auto Focus Functions

Detailed command is in below.

Step4. If reset or cut off power of OV5640, go to Step1.

16.4 How to use Embedded Auto Focus Solution

If you want to use AF module , please contact OmniVision local FAE for detail.

17. Frame exposure mode

17.1 Introduction

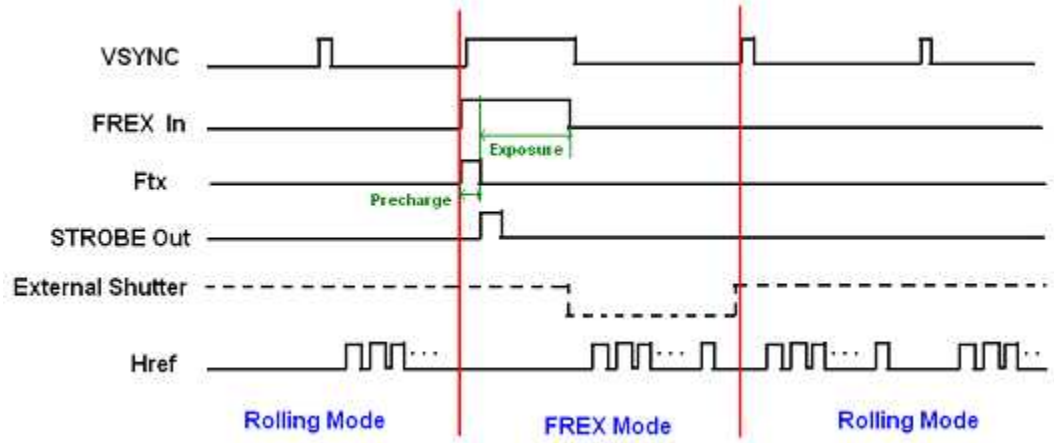
Compared with normal(rolling) exposure mode, frame exposure allows all rows integrated simultaneously rather than row by row. It is mainly used in image capture and can work accompany with flash strobe function to increase image brightness under dark situation.
Generally, the function works as follows:
(1) Initially, the sensor is in normal exposure mode. The electrical shutter is open.
(2) User sends a frex request.
(3) Photo-diode of all pixels in all rows start precharge and the precharge will last for a given time.
(4) Until a given precharge time, the precharge phase stops and all pixels start integration.
(5) During integration, flash strobe may turn on to increase exposure.
(6) Until a given exposure time, the electrical shutter closes to stop further integration, and whole image is then start to readout.
(7) After the readout is finished, the electrical shutter opens again and the sensor resume to normal mode.

17.2 FREX function mode

Since frex request can be sent from pad or from I2C, there are two FREX function mode:
(1) Mode 1: request from FREX input pad
(2) Mode 2: request from I2C.

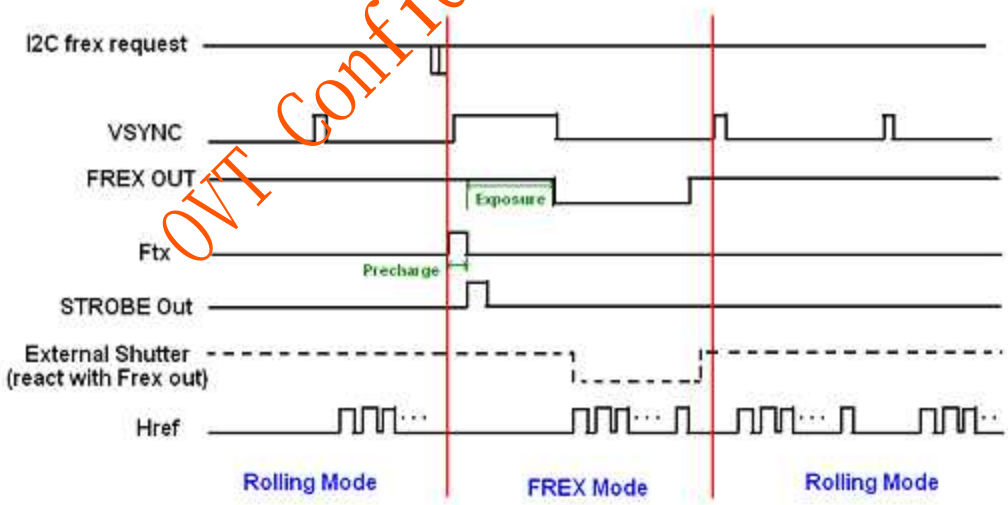
17.2.1 Pad mode

In mode 1, the FREX pad is used as the frex request input. STROBE pad is used as flash strobe signal output. The electrical shutter is controlled by system, not by sensor. The shutter signal should be turned On/Off according to the transition of FREX and VSYNC.
When frex request goes high, VSYNC goes high. The internal array control signal ftx and frst will go high to precharge all photo-diodes. STROBE out always starts at the end of ftx, and its width can be controlled by registers. The time interval between rising of STROBE and falling of FREX pad is the exposure time. By changing the frex request width, users can adjust the exposure time.
Frex request can come at any time of a frame. When it comes in the middle of a frame, the sensor stops output href/data. The current frame stops prematurely and each module starts the next frame freshly. VSYNC will go high after frex request goes high, and then keeps high until the request goes low.



17.2.2 I2c mode

In I2c mode , the frex request comes from I2C. Whenever users write 0x3b08 to 1, it sends one request pulse. The FREX pad is configured as output and is used as electrical shutter control signal. When frex request goes high, VSYNC goes high. The internal array control signal ftx and frst will go high to precharge all photo-diodes. After ftx goes low, integration starts. Unlike Mode1, the exposure time is controlled by registers. Shutter signal will low when exposure time is reached and will automatically resume high again after whole image is readout.



In both Mode 1 and Mode 2, we have the option(0x3817[3]) of turning off normal(rolling) mode timing. In that case, the sensor will not output image unless frex request comes. Every single request results in one single frame of image.

17.3 System settings

17.3.1 Pad mode

write_i2c(0x3016 ,0x02)
write_i2c(0x3017 ,0x7f)
write_i2c(0x3079 ,0x10)
write_i2c(0x471d ,0x02)
write_i2c(0x302c ,0x00)

17.3.2 I2c mode

write_i2c(0x3b07 ,0x09)
write_i2c(0x3016 ,0x02)
write_i2c(0x3017 ,0xff)
write_i2c(0x3709 ,0x10)
write_i2c(0x3b04 ,0x04)
write_i2c(0x3b05 ,0x00)

17.4 Option settings

17.4.1 Exposure Time (unit: Tline)

(a) 3b04: Exposure Time Highbyte default: 04
(b) 3b05: Exposure Time Lowbyte default: 00
The actual exposure time can be calculated by the equation:
(1 / System Frequency) * (Tline) * (Exposure Time in this setting)
Ex: If the system run 96Mhz, the actual exposure time is
(1/96000000) * ({380c, 380d}) * ({3b04, 3b05})

17.4.2 Frame Delay (unit: Tframe), 3b06[7:4] default: 0

Frame exposure can be set to start later, rather than starting right after the request coming.
When 3b06[7:4] is 0, frame exposure start right after frex request is coming.
When 3b06[7:4] is 1, frame exposure will wait for the end of the current frame to start.
When 3b06[7:4] is 2, frame exposure will wait 1 Tframe more than the previous case.

17.4.3 Flash Strobe Width (unit: Tline), 3b06[3:0] default: 4

Flash Strobe Pulse Width is (3b06[3:0] -1) Tlines

17.4.4 Frex Shutter Signal Reverse, 3b07[2] default: 0

1: reserse

0: not reserve (active low)

17.4.5 Frex Mode, 3b07[1:0] default: 0

1x: Frex Disable, rolling strobe mode
01: Frex Mode 2
00: Frex Mode 1

17.4.6 Frex Precharge Time(ftx, frst width), 3817[2:0] default: 4

0 : Tline/16
1 : Tline/8
2 : Tline/4
3 : Tline/2
5 : 2 * Tline
6 : 4 * Tline
7 : 8 * Tline
4 or 8: Tline

17.4.7 VSYNC ends time,

This setting determine how many lines VSYNC will go low after FREX input(Mode 1) or internal frex(Mode 2) goes low.
(a) 470a: VSYNC width adjust (unit: Tline) default: 04
(b) 370b: VSYNC width adjust (unit: Tpixel) default: 0

18. Some photos captured by 5640

Version History

Version 1.1 update brightness setting