OV5640 Auto Focus Camera Module Application Notes (with DVP Interface)

Last Modified: Nov. 4th, 2011

Document Revision: 2.13

内容目录

1. Overview of OV5640 Application....4
2. Hardware Design....5
2.1 OV5640 Camera Module Reference Design....5
2.2 Host Interface....6
2.1.1 Pin Definition....6
2.2 Power Supply....6
2.2.1 Host used 1.8V IO and OV5640 uses 1.8V IO....7
2.2.2 Host used 2.8V IO and OV5640 uses 1.8V IO....7
2.2.3 Host used 2.8V IO and OV5640 uses 1.8V IO....8
2.3 Deal with Lens....9
2.3.1 Light fall off....9
2.3.2 Dark corner....9
2.3.3 Resolution....9
2.3.4 Optical contrast....9
2.3.5 Lens Cover....9
2.3.6 Lens Correction....9
2.3.6.1 Lens Correction:....9
2.3.6.2 Lens Correction: 9
3. Hardware Operation....10
3.1 Operation Modes....10
3.1.1 Power Up....10
3.1.2 Power Down....11
3.1.3 Wake up from Power down....11
3.1.4 Power OFF 11
3.1.5 Hardware Reset....12
3.2 Operations....12
3.2.1 OV5640 single camera, cut of power when not used....12
3.2.2 OV5640 single camera, power down when not used....12
3.2.3 OV5640 with other camera, cut off power when not used....12
3.2.4 OV5640 with other camera, power down when not used....13
3.2.5 OV5640 share I2C bus with other devices....13
4. Software Operation....14
4.1 YCbCr/Compression Reference Settings....14
4.1.1 Initial Setting....14
4.1.2 VGA Preview....19
4.1.3 800x480 Preview....20
4.1.4 854x480 Preview....21
4.1.5 720p Video....23
4.1.6 5M YCbCr Capture....24
4.1.7 5M JPEG Capture....25
4.2 Signal Polarity....28
4.3 Drive Capability....28
4.4 I/O Control....28
4.5 YUV Sequence 29
4.6 Mirror and Flip....29
4.7 Test Pattern....31

4.8 Remove Light Band....31
4.9 User Interface Functions....32
4.9.1 Brightness....32
4.9.2 Contrast....33
4.9.3 Saturation....34
4.9.4 EV....37
4.9.5 Light Mode....38
4.9.6 Special Effects....39
4.9.7 Night Mode....41
4.7.8 Banding Filter Selection....41
4.10 Auto Focus....42
4.10.1 Embedded Auto Focus Solution....42
4.10.2 I2C Commands for Auto Focus....42
4.10.3 AF Sequence....43
4.10.4 Download firmware....43
4.10.5 Auto focus....43
4.10.6 Release Focus....43
4.11 Capture Sequence....43
4.11.1 Shutter....43
4.11.2 Gain....44
4.11.3 Dummy Lines and Dummy Pixels....44
4.11.4 Capture Sequence....44
4.11.4.1 Auto Focus....44
4.11.4.2 Read Preview Registers....44
4.11.4.3 Change Resolution to Capture....44
4.11.4.4 Read Capture Register Values....44
4.11.4.5 Preview Gain/Exposure to Capture/Gain Exposure....45
4.11.4.6 Gain to Exposure and Capture Banding Filter....45
4.11.4.7 Write gain/exposure value....45
4.11.4.8 Capture....46
4.11.4.9 Back to Preview....46
4.12 Scale and Zoom....46
4.12.1 Scale....46
4.12.2 Digital Zoom....47
Appendix I Sample Code of Camera Driver....49
Revision History....63

1. Overview of OV5640 Application

OV5640 is a 1/4 inch high performance 5M camera supporting both DVP and MIPI interface. This document focus on DVP interface application of OV5640. For MIPI interface application of OV5640, please read “OV5640 Camera Module Application Notes (with MIPI interface)”

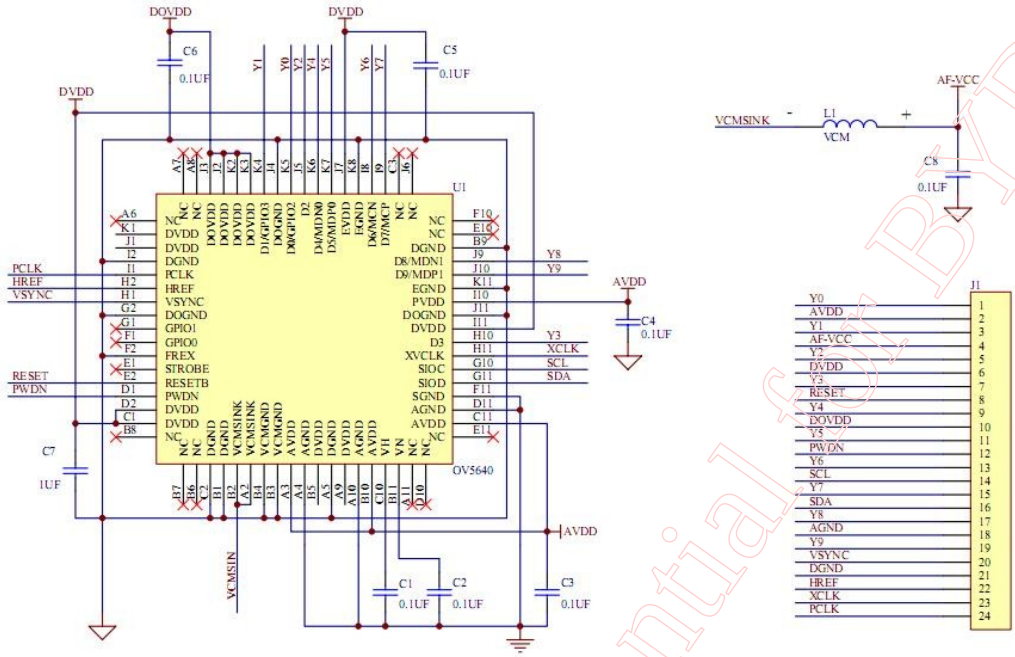
OV5640 could be used as

5M Main Camera for Cellular Phone Applications

5M Main Camera for Tablet Applications

2. Hardware Design

2.1 OV5640 Camera Module Reference Design



Note:

1. PWND, active HIGH as DOVDD to power down OV5640, should be connected to ground outside of module if unused
2. RESETB. Active LOW to reset OV5640, should be connected to DOVDD outside of module if unused
3. AVDD is 2.6-3.0V of sensor analog power (clean). 2.8V is recommended. AVDD must be 2.5V+-5% for OTP write, and OTP read does not have such requirement
4. DVDD is 1.5V±5% of sensor digital power(clean). Using the internal DVDD regulator is strongly recommended
5. DOVDD. 1.8V recommended is 1.7V-3.0V of sensor digital IO power(clean)
6. sensor AGND and DGND should be separated and connected to a single point outside PCB, Do not connect inside module
7. Capacitors should be close to the related sensor pins
8. D[9:0] is sensor 10 bit RGB RAW output. D[9:2] is sensor 8-bit YUV/RGB Compress output

2.2 Host Interface

2.1.1 Pin Definition

The video port of OV5640 has 10-bit, D[9:0]. For 10-bit RGB raw output, please use D[9:0]. For 8-bit YCbCr or 8-bit RGB raw or 8-bit RGB 565 output, please use D[9:2].

The Href and Hsync signal is on the same pin – Href. The function of this pin could be selected by SCCB setting.

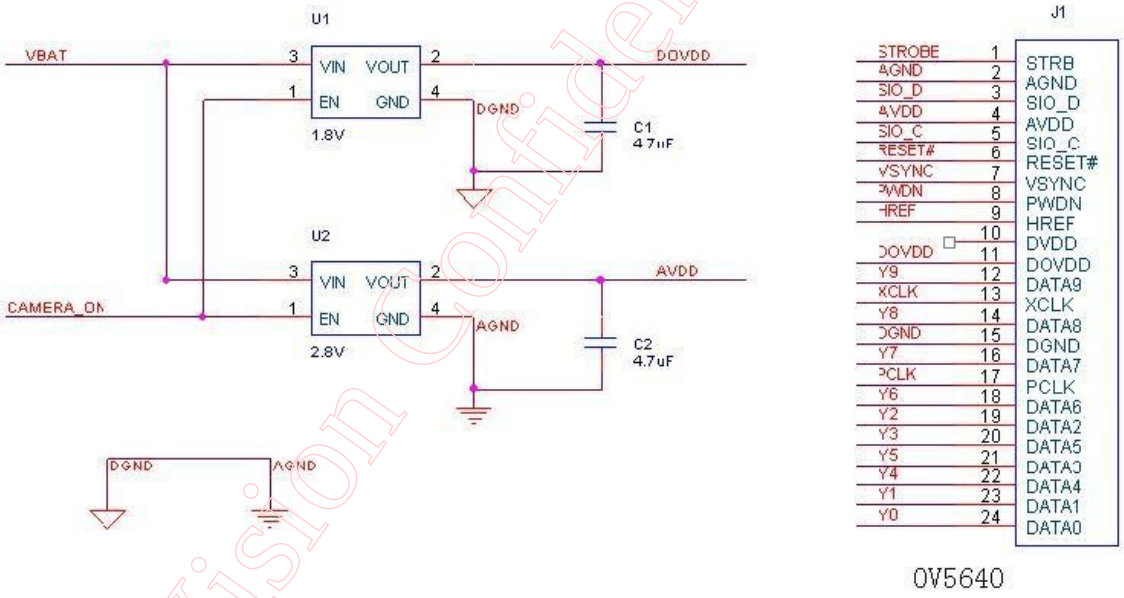
The SIO_C and SIO_D bus should have external pull up resistors, the typical value of the pull up resistors is 5.1K.

RESET# is active low with internal pull-up resistor. Reset# should be controlled by backend chip for proper power up sequence.

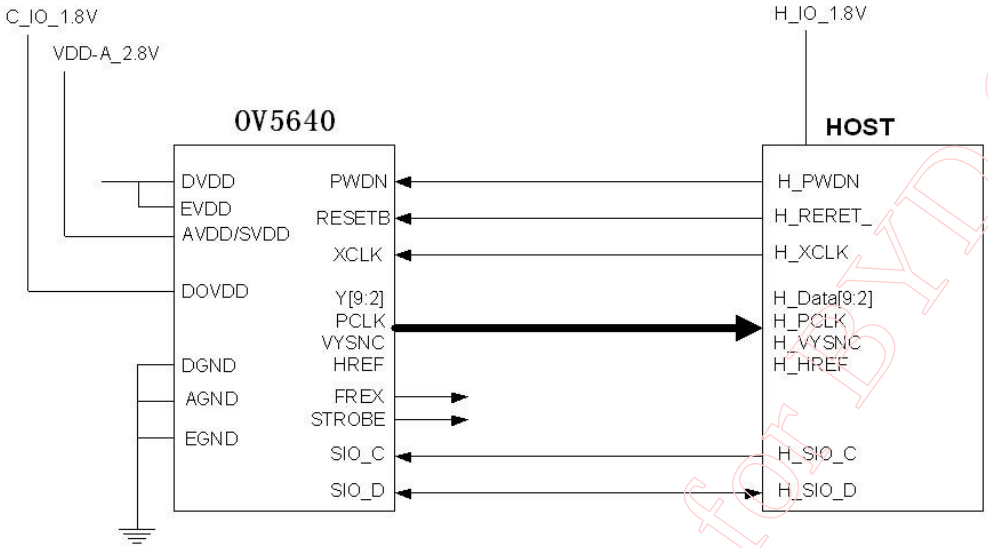
PWDN is active high with internal pull-down resistor. PWDN should be controlled by backend chip for proper power up sequence.

2.2 Power Supply

If DOVDD of OV5650 is 1.8V, then DVDD is generated by internal regulator of OV5640. So 2 regulators should be used to supply powers to OV5640.

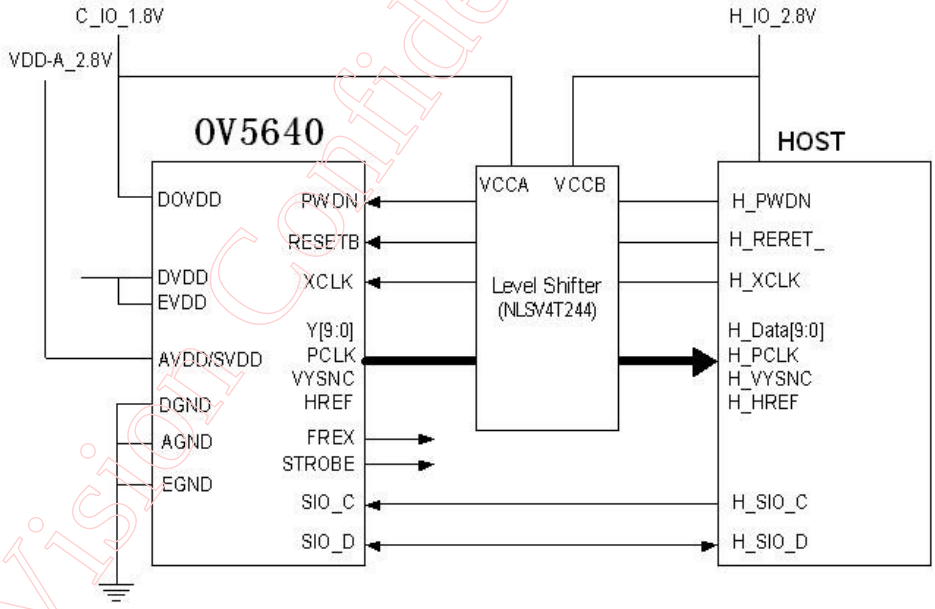


2.2.1 Host used 1.8V IO and OV5640 uses 1.8V IO



2.2.2 Host used 2.8V IO and OV5640 uses 1.8V IO

However, for some legacy phones or baseband chips, they only accepts 2.8V IO signals, for this kind of application, we suggest 5640 use 1.8V with level shift in between 5640 and host for all video signals.



In this case, 5640 works in the 1.8V IO, and host works in required 2.8V IO. For this case:

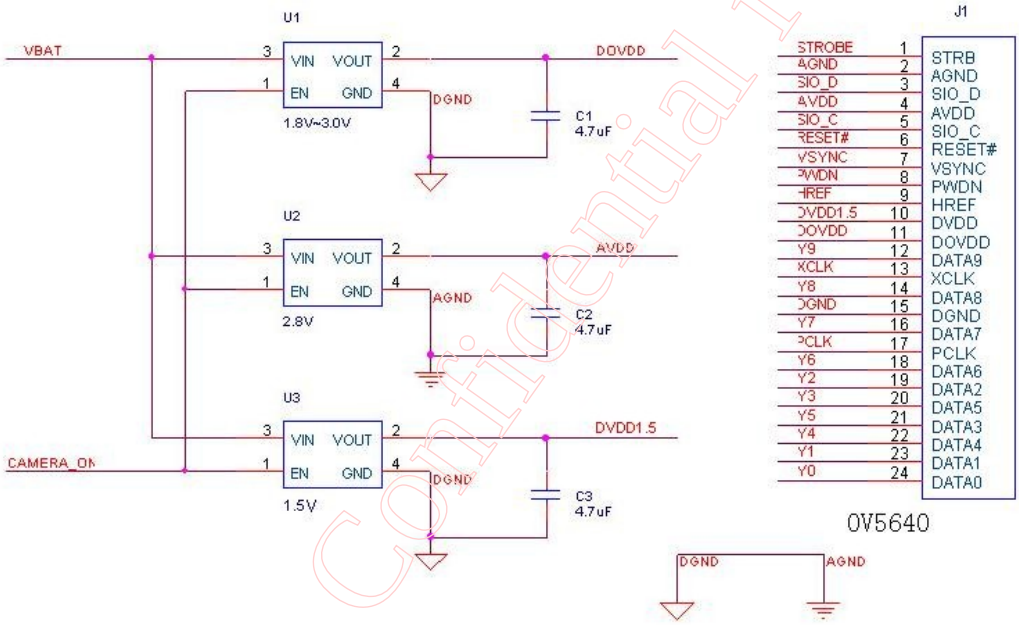
1) OV5640 output pins (HREF/HSYNC, VSYNC, DATA0~9, PCLK, FREX, STROBE, GPIO1~2) are tri-state when sensor is in standby mode (PWDN pin is high). Please make sure that it would not cause any problem(current leak) for other devices' input. You may put 10kohm pull-down resistor on all tri-state pins if tri-state is an issue for level-shifter or host

chip.

2) Add level-shifter for all video signals -- HREF/HSYNC, VSYNC, DATA0~9, PCLK.
3) Sensor SCL and SDA can tolerance high voltage. So it is no problem for 5640 to use 1.8V IO and I2C pull up resistors connect to 2.8V.
4) Sensor PWDN and XCLK are no problem for 2.8V signal. PWDN can directly connect to 2.8V for sensor power down. If has extra level shift pins, add level shift for these two pins as well.
5) For RESETB pin, please don't provide 2.8V signal since 5640 has internal pull-up resistor to IO power. If provide 2.8V signal on this pin, it will cause power leak (from 2.8V signal to the sensor 1.8V IO power through the pull-up resistor and diode). Here are some solutions.

a) Host drive this pin to low for reset mode, and change it to floating/input in operating mode and let sensor internal pull-up resistor to drive high (1.8V).
b) Add level-shifter for RESETB as well.

If DOVDD of OV5640 is 2.8V, then DVDD is supplied from external power supply. So 3 regulators should be used. Power down mode is not supported for power saving here.



2.2.3 Host used 2.8V IO and OV5640 uses 1.8V IO

OV5640 also support 2.8V IO. If use 2.8V IO, the 1.5V DVDD must be supplied by external power. And power off mode is recommended for power saving. Power down mode is not supported for power saving.

2.3 Deal with Lens

2.3.1 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.

2.3.2 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.

2.3.3 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.

2.3.4 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.

2.3.5 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.

2.3.6 Lens Correction

Lens Correction setting should be tuned for every module. Please find lens correction settings of modules tuned by OVT FAE.

2.3.6.1 Lens Correction:

2.3.6.2 Lens Correction:

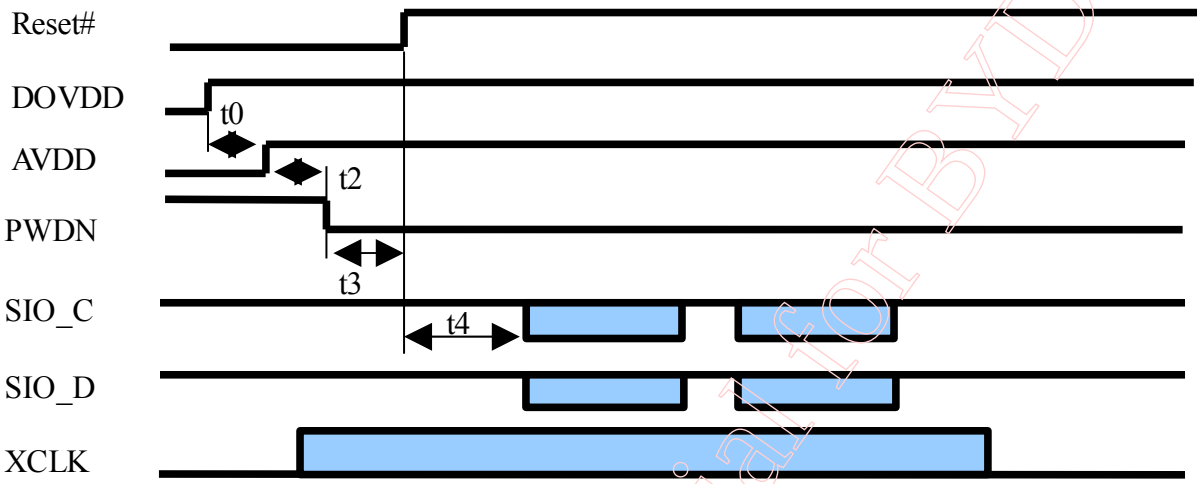
Note:

If module/lens you are using can not be found above, please contact with OmniVision local FAE for lens correction settings.

3. Hardware Operation

3.1 Operation Modes

3.1.1 Power Up



t0: >= 0ms. Delay from DOVDD stable to AVDD stable.
t2: >= 5ms. Delay from AVDD stable to sensor power up stable.
t3: >= 1ms. Delay from sensor power up stable to Reset# pull high.
t4: >=20ms. Delay from Reset pull high to SCCB initialization.

Step 1:
Reset# is applied to OV5640 camera module. PWDN is pulled high.

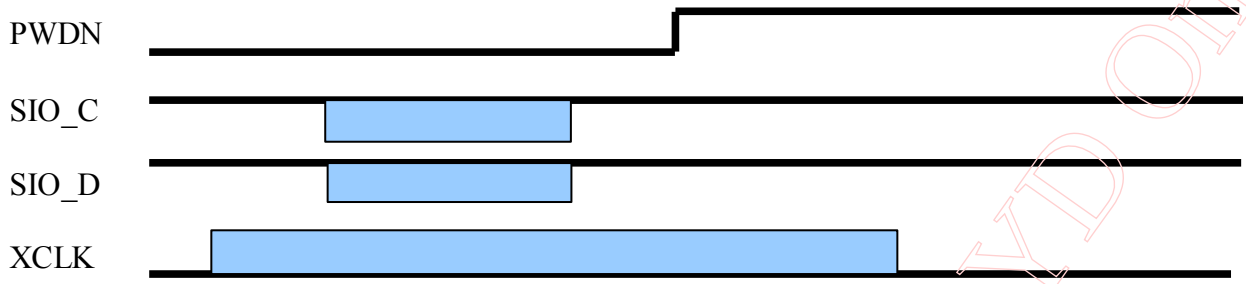
Step 2:
DOVDD and AVDD powers are applied. The 2 powers could be applied simultaneously. If applied separately, the power on sequence should be DOVDD first, and AVDD last.

Step 3:
after 5ms of AVDD reaching stable, pull PWDN to low.

Step 4:
after 1ms of PWDN go low, pull high Reset#.

Step 5:
After 20ms, initialize OV5640 by SCCB initialization. Please find the settings from section ?-?.

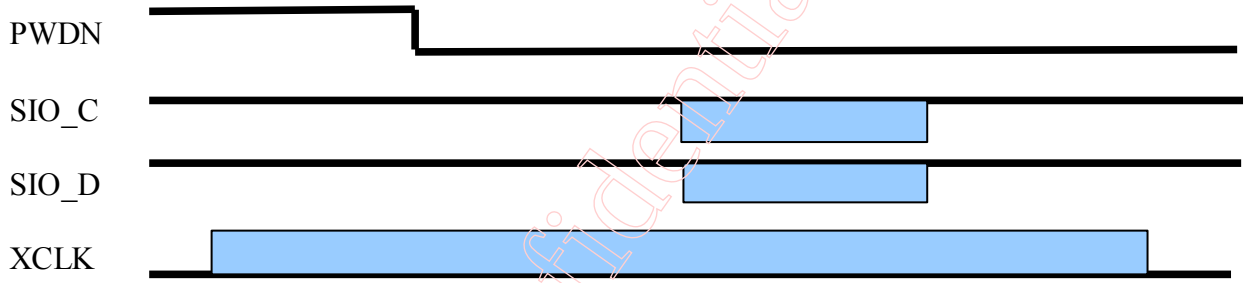
3.1.2 Power Down



Step 1:
Pull PWDN pin high.

Step 2:
Pull XCLK low. XCLK should be keep more than 0.1ms after PWDN pulled high.

3.1.3 Wake up from Power down



Step 1:
Apply XCLK

Step 2:
after 0.1ms, Pull low PWDN

Optional Step 3:
Full SCCB Initialization. Please find the settings from section ?-?.

3.1.4 Power OFF

Step 1.
Pull low XCLK,

Step 2.
Turn off AVDD, DVDD and DOVDD. The 3 powers could be turned off simultaneously. If turned off separately, DVDD should be turned off first, AVDD second and DOVDD third.

Step 3.

Pull Low PWDN and RESET

3.1.5 Hardware Reset

OV5640 sensor includes a RESETB pin that forces a complete hardware reset when it is pulled low(GND). OV5640 clears all registers and resets them to their default values when a hardware reset occurred. A reset can also be initiated through the SCCB interface by setting register 0x12[7] to high.

The whole chip will be reset during power up. Manually applying a hard reset after power up is recommended even through the on-chip power up reset is included. The hard reset is active low with an asynchronous design. The reset pulse width should be greater than or equal to 1ms.

3.2 Operations

3.2.1 OV5640 single camera, cut of power when not used

Mode	Operation
Battery On	No
Enter Camera	Power On Hardware Reset Initialization
Exit Camera	Power Off

3.2.2 OV5640 single camera, power down when not used

Mode	
Battery On	Power Up Hardware Reset Power Down
Enter Camera	Wake up from Power Down Initialization
Exit From Camera	Power Down

3.2.3 OV5640 with other camera, cut off power when not used

Mode	Operation
Battery On	No
Enter OV5640	Power up Both, Keep PWDN of other camera high, PWDN of OV5640 low Hardware Reset Initialize OV5640

Enter Other Camera	Power up Both, Keep PWDN of OV5640 high, PWDN of other camera low Hardware Reset Initialize other camera
Switch from OV5640 to Other Camera	Power down OV5640 Wake up other camera Initialize other camera
Switch from Other Camera to OV5640	Power down other camera Wake up OV5640 Initialize OV5640
Exit both Camera	Power Off Both

3.2.4 OV5640 with other camera, power down when not used

Mode	Operation
Battery On	Power up Both, Keep PWDN of other camera low, PWDN of OV5640 high Hardware Reset Initialize other camera Pull high PWDN of other camera, Pull low PWDN of OV5640 Initialize OV5640 Pull high PWDN of OV5640
Enter OV5640	Wake up OV5640 Initialize OV5640
Exit OV5640	Power down OV5640
Enter Other Camera	Wake up other camera Initialize other camera
Exit Other Camera	Power down other camera

3.2.5 OV5640 share I2C bus with other devices

States other devices access I^2C	States other devices can not access I^2C
Power Off, Power Down, Wake up from Power Down Power up	None

4. Software Operation

4.1 YCbCr/Compression Reference Settings

4.1.1 Initial Setting

```
//15fps VGA YUV output
// 24MHz input clock, 24MHz PCLK
write_i2c(0x3103, 0x11);        // system clock from pad, bit[1]
write_i2c(0x3008, 0x82);        // software reset, bit[7]
// delay 5ms
write_i2c(0x3008, 0x42);        // software power down, bit[6]
write_i2c(0x3103, 0x03);        // system clock from PLL, bit[1]
write_i2c(0x3017, 0xff);        // FREX, Vsync, HREF, PCLK, D[9:6] output enable
write_i2c(0x3018, 0xff);        // D[5:0], GPIO[1:0] output enable
write_i2c(0x3034, 0x1a);        // MIPI 10-bit
write_i2c(0x3037, 0x13);        // PLL root divider, bit[4], PLL pre-divider, bit[3:0]
write_i2c(0x3108, 0x01);        // PCLK root divider, bit[5:4], SCLK2x root divider, bit[3:2]
                                // SCLK root divider, bit[1:0]
write_i2c(0x3630, 0x36);
write_i2c(0x3631, 0x0e);
write_i2c(0x3632, 0xe2);
write_i2c(0x3633, 0x12);
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);        // VCM control
write_i2c(0x3601, 0x33);        // VCM control
write_i2c(0x302d, 0x60);        // system control
write_i2c(0x3620, 0x52);
write_i2c(0x371b, 0x20);
write_i2c(0x471c, 0x50);
write_i2c(0x3a13, 0x43);        // pre-gain = 1.047x
write_i2c(0x3a18, 0x00);        // gain ceiling
write_i2c(0x3a19, 0xf8);        // gain ceiling = 15.5x
write_i2c(0x3635, 0x13);
write_i2c(0x3636, 0x03);
write_i2c(0x3634, 0x40);
write_i2c(0x3622, 0x01);
```

```
// 50/60Hz detection
write_i2c(0x3c01, 0x34);        // Band auto, bit[7]
write_i2c(0x3c04, 0x28);        // threshold low sum
write_i2c(0x3c05, 0x98);        // threshold high sum
write_i2c(0x3c06, 0x00);        // light meter 1 threshold[15:8]
write_i2c(0x3c07, 0x08);        // light meter 1 threshold[7:0]
write_i2c(0x3c08, 0x00);        // light meter 2 threshold[15:8]
write_i2c(0x3c09, 0x1c);        // light meter 2 threshold[7:0]
write_i2c(0x3c0a, 0x9c);        // sample number[15:8]
write_i2c(0x3c0b, 0x40);        // sample number[7:0]

write_i2c(0x3810, 0x00);        // Timing Hoffset[11:8]
write_i2c(0x3811, 0x10);        // Timing Hoffset[7:0]
write_i2c(0x3812, 0x00);        // Timing Voffset[10:8]
write_i2c(0x3708, 0x64);
write_i2c(0x4001, 0x02);        // BLC start from line 2
write_i2c(0x4005, 0x1a);        // BLC always update
write_i2c(0x3000, 0x00);        // enable blocks
write_i2c(0x3004, 0xff);        // enable clocks
write_i2c(0x300e, 0x58);        // MIPI power down, DVP enable
write_i2c(0x302e, 0x00);
write_i2c(0x4300, 0x30);        // YUV 422, YUYV
write_i2c(0x501f, 0x00);        // YUV 422
write_i2c(0x440e, 0x00);
write_i2c(0x5000, 0xa7);        // Lenc on, raw gamma on, BPC on, WPC on, CIP on

// AEC target
write_i2c(0x3a0f, 0x30);        // stable range in high
write_i2c(0x3a10, 0x28);        // stable range in low
write_i2c(0x3a1b, 0x30);        // stable range out high
write_i2c(0x3a1e, 0x26);        // stable range out low
write_i2c(0x3a11, 0x60);        // fast zone high
write_i2c(0x3a1f, 0x14);        // fast zone low

// Lens correction for ?
write_i2c(0x5800, 0x23);
write_i2c(0x5801, 0x14);
write_i2c(0x5802, 0x0f);
write_i2c(0x5803, 0x0f);
write_i2c(0x5804, 0x12);
write_i2c(0x5805, 0x26);
write_i2c(0x5806, 0x0c);
write_i2c(0x5807, 0x08);
write_i2c(0x5808, 0x05);
write_i2c(0x5809, 0x05);
write_i2c(0x580a, 0x08);
write_i2c(0x580b, 0x0d);
```

```
write_i2c(0x580c, 0x08);
write_i2c(0x580d, 0x03);
write_i2c(0x580e, 0x00);
write_i2c(0x580f, 0x00);
write_i2c(0x5810, 0x03);
write_i2c(0x5811, 0x09);
write_i2c(0x5812, 0x07);
write_i2c(0x5813, 0x03);
write_i2c(0x5814, 0x00);
write_i2c(0x5815, 0x01);
write_i2c(0x5816, 0x03);
write_i2c(0x5817, 0x08);
write_i2c(0x5818, 0x0d);
write_i2c(0x5819, 0x08);
write_i2c(0x581a, 0x05);
write_i2c(0x581b, 0x06);
write_i2c(0x581c, 0x08);
write_i2c(0x581d, 0x0e);
write_i2c(0x581e, 0x29);
write_i2c(0x581f, 0x17);
write_i2c(0x5820, 0x11);
write_i2c(0x5821, 0x11);
write_i2c(0x5822, 0x15);
write_i2c(0x5823, 0x28);
write_i2c(0x5824, 0x46);
write_i2c(0x5825, 0x26);
write_i2c(0x5826, 0x08);
write_i2c(0x5827, 0x26);
write_i2c(0x5828, 0x64);
write_i2c(0x5829, 0x26);
write_i2c(0x582a, 0x24);
write_i2c(0x582b, 0x22);
write_i2c(0x582c, 0x24);
write_i2c(0x582d, 0x24);
write_i2c(0x582e, 0x06);
write_i2c(0x582f, 0x22);
write_i2c(0x5830, 0x40);
write_i2c(0x5831, 0x42);
write_i2c(0x5832, 0x24);
write_i2c(0x5833, 0x26);
write_i2c(0x5834, 0x24);
write_i2c(0x5835, 0x22);
write_i2c(0x5836, 0x22);
write_i2c(0x5837, 0x26);
write_i2c(0x5838, 0x44);
write_i2c(0x5839, 0x24);
write_i2c(0x583a, 0x26);
write_i2c(0x583b, 0x28);
```

```
write_i2c(0x583c, 0x42);
write_i2c(0x583d, 0xce);        // lenc BR offset

// AWB
write_i2c(0x5180, 0xff);        // AWB B block
write_i2c(0x5181, 0xf2);        // AWB control
write_i2c(0x5182, 0x00);        // [7:4] max local counter, [3:0] max fast counter
write_i2c(0x5183, 0x14);        // AWB advanced
write_i2c(0x5184, 0x25);
write_i2c(0x5185, 0x24);
write_i2c(0x5186, 0x09);
write_i2c(0x5187, 0x09);
write_i2c(0x5188, 0x09);
write_i2c(0x5189, 0x75);
write_i2c(0x518a, 0x54);
write_i2c(0x518b, 0xe0);
write_i2c(0x518c, 0xb2);
write_i2c(0x518d, 0x42);
write_i2c(0x518e, 0x3d);
write_i2c(0x518f, 0x56);
write_i2c(0x5190, 0x46);
write_i2c(0x5191, 0xf8);        // AWB top limit
write_i2c(0x5192, 0x04);        // AWB bottom limit
write_i2c(0x5193, 0x70);        // red limit
write_i2c(0x5194, 0xf0);        // green limit
write_i2c(0x5195, 0xf0);        // blue limit
write_i2c(0x5196, 0x03);        // AWB control
write_i2c(0x5197, 0x01);        // local limit
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);        // AWB control

// Gamma
write_i2c(0x5480, 0x01);        // Gamma bias plus on, bit[0]
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);

// color matrix
write_i2c(0x5381, 0x1e);      // CMX1 for Y
write_i2c(0x5382, 0x5b);      // CMX2 for Y
write_i2c(0x5383, 0x08);      // CMX3 for Y
write_i2c(0x5384, 0x0a);      // CMX4 for U
write_i2c(0x5385, 0x7e);      // CMX5 for U
write_i2c(0x5386, 0x88);      // CMX6 for U
write_i2c(0x5387, 0x7c);      // CMX7 for V
write_i2c(0x5388, 0x6c);      // CMX8 for V
write_i2c(0x5389, 0x10);      // CMX9 for V
write_i2c(0x538a, 0x01);      // sign[9]
write_i2c(0x538b, 0x98);      // sign[8:1]

// UV adjsut
write_i2c(0x5580, 0x06);      // saturation on, bit[1]
write_i2c(0x5583, 0x40);
write_i2c(0x5584, 0x10);
write_i2c(0x5589, 0x10);
write_i2c(0x558a, 0x00);
write_i2c(0x558b, 0xf8);
write_i2c(0x501d, 0x40);      // enable manual offset of contrast

// CIP
write_i2c(0x5300, 0x08);      // CIP sharpen MT threshold 1
write_i2c(0x5301, 0x30);      // CIP sharpen MT threshold 2
write_i2c(0x5302, 0x10);      // CIP sharpen MT offset 1
write_i2c(0x5303, 0x00);      // CIP sharpen MT offset 2
write_i2c(0x5304, 0x08);      // CIP DNS threshold 1
write_i2c(0x5305, 0x30);      // CIP DNS threshold 2
write_i2c(0x5306, 0x08);      // CIP DNS offset 1
write_i2c(0x5307, 0x16);      // CIP DNS offset 2
write_i2c(0x5309, 0x08);      // CIP sharpen TH threshold 1
write_i2c(0x530a, 0x30);      // CIP sharpen TH threshold 2
write_i2c(0x530b, 0x04);      // CIP sharpen TH offset 1
write_i2c(0x530c, 0x06);      // CIP sharpen TH offset 2

write_i2c(0x5025, 0x00);
write_i2c(0x3008, 0x02);      // wake up from standby, bit[6]
```

4.1.2 VGA Preview

VGA 30fps

```
// YUV VGA 30fps, night mode 5fps
// Input Clock = 24Mhz, PCLK = 56MHz
write_i2c(0x3035, 0x11);        // PLL
write_i2c(0x3036, 0x46);        // PLL
write_i2c(0x3c07, 0x08);        // light meter 1 threshold [7:0]
write_i2c(0x3820, 0x41);        // Sensor flip off, ISP flip on
write_i2c(0x3821, 0x07);        // Sensor mirror on, ISP mirror on, H binning on

write_i2c(0x3814, 0x31);        // X INC
write_i2c(0x3815, 0x31);        // Y INC
write_i2c(0x3800, 0x00);        // HS
write_i2c(0x3801, 0x00);        // HS
write_i2c(0x3802, 0x00);        // VS
write_i2c(0x3803, 0x04);        // VS
write_i2c(0x3804, 0x0a);        // HW (HE)
write_i2c(0x3805, 0x3f);        // HW (HE)
write_i2c(0x3806, 0x07);        // VH (VE)
write_i2c(0x3807, 0x9b);        // VH (VE)
write_i2c(0x3808, 0x02);        // DVPHO
write_i2c(0x3809, 0x80);        // DVPHO
write_i2c(0x380a, 0x01);        // DVPVO
write_i2c(0x380b, 0xe0);        // DVPVO
write_i2c(0x380c, 0x07);        // HTS
write_i2c(0x380d, 0x68);        // HTS
write_i2c(0x380e, 0x03);        // VTS
write_i2c(0x380f, 0xd8);        // VTS
write_i2c(0x3813, 0x06);        // Timing Voffset

write_i2c(0x3618, 0x00);
write_i2c(0x3612, 0x29);
write_i2c(0x3709, 0x52);
write_i2c(0x370c, 0x03);

write_i2c(0x3a02, 0x17);        // 60Hz max exposure, night mode 5fps
write_i2c(0x3a03, 0x10);        // 60Hz max exposure
// banding filters are calculated automatically in camera driver
//write_i2c(0x3a08, 0x01);          // B50 step
//write_i2c(0x3a09, 0x27);          // B50 step
//write_i2c(0x3a0a, 0x00);          // B60 step
//write_i2c(0x3a0b, 0xf6);          // B60 step
//write_i2c(0x3a0e, 0x03);          // 50Hz max band
//write_i2c(0x3a0d, 0x04);          // 60Hz max band
write_i2c(0x3a14, 0x17);        // 50Hz max exposure, night mode 5fps
write_i2c(0x3a15, 0x10);        // 50Hz max exposure
write_i2c(0x4004, 0x02);        // BLC 2 lines
write_i2c(0x3002, 0x1c);        // reset JFIFO, SFIFO, JPEG
```

```
write_i2c(0x3006, 0xc3);        // disable clock of JPEG2x, JPEG
write_i2c(0x4713, 0x03);        // JPEG mode 3
write_i2c(0x4407, 0x04);        // Quantization scale
write_i2c(0x460b, 0x35);
write_i2c(0x460c, 0x22);
write_i2c(0x4837, 0x22);        // DVP CLK divider
write_i2c(0x3824, 0x02);        // DVP CLK divider
write_i2c(0x5001, 0xa3);        // SDE on, scale on, UV average off, color matrix on, AWB on
write_i2c(0x3503, 0x00);        // AEC/AGC on
```

VGA 15fps

```
// YUV VGA 15fps, night mode 5fps
// Input Clock = 24Mhz, PCLK = 28MHz
//same settings as VGA 30fps, except the following settings
write_i2c(0x3035, 0x21);        // PLL
write_i2c(0x3a02, 0x0b);        // 60Hz max exposure, night mode 5fps
write_i2c(0x3a03, 0x88);        // 60Hz max exposure
write_i2c(0x3a14, 0x0b);        // 50Hz max exposure, night mode 5fps
write_i2c(0x3a15, 0x88);        // 50Hz max exposure
```

4.1.3 800x480 Preview

800x480 15fps

```
// 800x480 15fps, night mode 5fps
// input clock 24Mhz, PCLK 45.6Mhz
write_i2c(0x3035, 0x41);        // PLL
write_i2c(0x3036, 0x72);        // PLL
write_i2c(0x3c07, 0x08);        // light meter 1 threshold[7:0]
write_i2c(0x3820, 0x41);        // flip
write_i2c(0x3821, 0x07);        // mirror
write_i2c(0x3814, 0x31);        // timing X inc
write_i2c(0x3815, 0x31);        // timing Y inc
write_i2c(0x3800, 0x00);        // HS
write_i2c(0x3801, 0x00);        // HS
write_i2c(0x3802, 0x00);        // VS
write_i2c(0x3803, 0xbe);        // VS
write_i2c(0x3804, 0x0a);        // HW (HE)
write_i2c(0x3805, 0x3f);        // HW (HE)
write_i2c(0x3806, 0x06);        // VH (VE)
write_i2c(0x3807, 0xe4);        // VH (VE)
write_i2c(0x3808, 0x03);        // DVPHO
write_i2c(0x3809, 0x20);        // DVPHO
write_i2c(0x380a, 0x01);        // DVPVO
write_i2c(0x380b, 0xe0);        // DVPVO
write_i2c(0x380c, 0x07);        // HTS
write_i2c(0x380d, 0x69);        // HTS
write_i2c(0x380e, 0x03);        // VTS
write_i2c(0x380f, 0x21);        // VTS
```

```
write_i2c(0x3813, 0x06);        // timing V offset
write_i2c(0x3618, 0x00);
write_i2c(0x3612, 0x29);
write_i2c(0x3709, 0x52);
write_i2c(0x370c, 0x03);
write_i2c(0x3a02, 0x09);        // 60Hz max exposure, night mode 5fps
write_i2c(0x3a03, 0x63);        // 60Hz max exposure
// banding filters are calculated automatically in camera driver
//write_i2c(0x3a08, 0x00);          // B50 step
//write_i2c(0x3a09, 0x78);          // B50 step
//write_i2c(0x3a0a, 0x00);          // B60 step
//write_i2c(0x3a0b, 0x64);          // B60 step
//write_i2c(0x3a0e, 0x06);          // 50Hz max band
//write_i2c(0x3a0d, 0x08);          // 60Hz max band
write_i2c(0x3a14, 0x09);        // 50Hz max exposure, night mode 5fps
write_i2c(0x3a15, 0x63);        // 50Hz max exposure
write_i2c(0x4004, 0x02);        // BLC line number
write_i2c(0x3002, 0x1c);        // reset JFIFO, SFIFO, JPG
write_i2c(0x3006, 0xc3);        // disable clock of JPEG2x, JPEG
write_i2c(0x4713, 0x03);        // JPEG mode 3
write_i2c(0x4407, 0x04);        // Quantization sacle
write_i2c(0x460b, 0x35);
write_i2c(0x460c, 0x22);
write_i2c(0x4837, 0x22);        // MIPI global timing
write_i2c(0x3824, 0x02);        // PCLK manual divider
write_i2c(0x5001, 0xa3);        // SDE on, CMX on, AWB on
write_i2c(0x3503, 0x00);        // AEC/AGC on
```

800x480 30fps

```
// YUV 800x480 30fps, night mode 5fps
// Input Clock = 24Mhz, PCLK = 91.2MHz
//same settings as 800x480 15fps, except the following settings
write_i2c(0x3035, 0x21);        // PLL
write_i2c(0x3a02, 0x12);        // 60Hz max exposure, night mode 5fps
write_i2c(0x3a03, 0xc6);        // 60Hz max exposure
write_i2c(0x3a14, 0x12);        // 50Hz max exposure, night mode 5fps
write_i2c(0x3a15, 0xc6);        // 50Hz max exposure
```

4.1.4 854x480 Preview

854x480 15fps

```
// 854x480 15fps, night mode 5fps
// input clock 24Mhz, PCLK 42Mhz
write_i2c(0x3035, 0x41);        // PLL
write_i2c(0x3036, 0x69);        // PLL
write_i2c(0x3c07, 0x07);        // light meter 1 threshold[7:0]
write_i2c(0x3820, 0x41);        // flip
write_i2c(0x3821, 0x07);        // mirror
```

```
write_i2c(0x3814, 0x31);        // timing X inc
write_i2c(0x3815, 0x31);        // timing Y inc
write_i2c(0x3800, 0x00);        // HS
write_i2c(0x3801, 0x00);        // HS
write_i2c(0x3802, 0x00);        // VS
write_i2c(0x3803, 0xfa);        // VS
write_i2c(0x3804, 0x0a);        // HW (HE)
write_i2c(0x3805, 0x3f);        // HW (HE)
write_i2c(0x3806, 0x06);        // VH (VE)
write_i2c(0x3807, 0xa9);        // VH (VE)
write_i2c(0x3808, 0x03);        // DVPHO
write_i2c(0x3809, 0x56);        // DVPHO
write_i2c(0x380a, 0x01);        // DVPVO
write_i2c(0x380b, 0xe0);        // DVPVO
write_i2c(0x380c, 0x07);        // HTS
write_i2c(0x380d, 0x64);        // HTS
write_i2c(0x380e, 0x02);        // VTS
write_i2c(0x380f, 0xe4);        // VTS
write_i2c(0x3813, 0x04);        // timing V offset
write_i2c(0x3618, 0x00);
write_i2c(0x3612, 0x29);
write_i2c(0x3709, 0x52);
write_i2c(0x370c, 0x03);
write_i2c(0x3a02, 0x08);        // 60Hz max exposure, night mode 5fps
write_i2c(0x3a03, 0xac);        // 60Hz max exposure
// banding filters are calculated automatically in camera driver
//write_i2c(0x3a08, 0x00);          // B50 step
//write_i2c(0x3a09, 0x6f);          // B50 step
//write_i2c(0x3a0a, 0x00);          // B60 step
//write_i2c(0x3a0b, 0x5c);          // B60 step
//write_i2c(0x3a0e, 0x06);          // 50Hz max band
//write_i2c(0x3a0d, 0x08);          // 60Hz max band
write_i2c(0x3a14, 0x08);        // 50Hz max exposure, night mode 5fps
write_i2c(0x3a15, 0xac);        // 50Hz max exposure
write_i2c(0x4004, 0x02);        // BLC line number
write_i2c(0x3002, 0x1c);        // reset JFIFO, SFIFO, JPG
write_i2c(0x3006, 0xc3);        // disable clock of JPEG2x, JPEG
write_i2c(0x4713, 0x03);        // JPEG mode 3
write_i2c(0x4407, 0x04);        // Quantization sacle
write_i2c(0x460b, 0x37);
write_i2c(0x460c, 0x20);
write_i2c(0x4837, 0x16);        // MIPI global timing
write_i2c(0x3824, 0x04);        // PCLK manual divider
write_i2c(0x5001, 0xa3);        // SDE on, scale on, CMX on, AWB on
write_i2c(0x3503, 0x00);        // AEC/AGC on
```

854x480 30fps

```
// YUV 854x480 30fps, night mode 5fps
```

```
// Input Clock = 24Mhz, PCLK = 84MHz
//same settings as 854x480 15fps, except the following settings
write_i2c(0x3035, 0x21);        // PLL
write_i2c(0x3a02, 0x11);        // 60Hz max exposure, night mode 5fps
write_i2c(0x3a03, 0x58);        // 60Hz max exposure
write_i2c(0x3a14, 0x11);        // 50Hz max exposure, night mode 5fps
write_i2c(0x3a15, 0x58);        // 50Hz max exposure
```

4.1.5 720p Video

720p 15fps

```
// 1280x720, 15fps
// input clock 24Mhz, PCLK 42Mhz
write_i2c(0x3035, 0x41);        // PLL
write_i2c(0x3036, 0x69);        // PLL
write_i2c(0x3c07, 0x07);        // lightmeter 1 threshold[7:0]
write_i2c(0x3820, 0x41);        // flip
write_i2c(0x3821, 0x07);        // mirror
write_i2c(0x3814, 0x31);        // timing X inc
write_i2c(0x3815, 0x31);        // timing Y inc
write_i2c(0x3800, 0x00);        // HS
write_i2c(0x3801, 0x00);        // HS
write_i2c(0x3802, 0x00);        // VS
write_i2c(0x3803, 0xfa);        // VS
write_i2c(0x3804, 0x0a);        // HW (HE)
write_i2c(0x3805, 0x3f);        // HW (HE)
write_i2c(0x3806, 0x06);        // VH (VE)
write_i2c(0x3807, 0xa9);        // VH (VE)
write_i2c(0x3808, 0x05);        // DVPHO
write_i2c(0x3809, 0x00);        // DVPHO
write_i2c(0x380a, 0x02);        // DVPVO
write_i2c(0x380b, 0xd0);        // DVPVO
write_i2c(0x380c, 0x07);        // HTS
write_i2c(0x380d, 0x64);        // HTS
write_i2c(0x380e, 0x02);        // VTS
write_i2c(0x380f, 0xe4);        // VTS
write_i2c(0x3813, 0x04);        // timing V offset
write_i2c(0x3618, 0x00);
write_i2c(0x3612, 0x29);
write_i2c(0x3709, 0x52);
write_i2c(0x370c, 0x03);
write_i2c(0x3a02, 0x02);        // 60Hz max exposure
write_i2c(0x3a03, 0xe0);        // 60Hz max exposure
// banding filters are calculated automatically in camera driver
//write_i2c(0x3a08, 0x00);          // B50 step
//write_i2c(0x3a09, 0x6f);          // B50 step
//write_i2c(0x3a0a, 0x00);          // B60 step
//write_i2c(0x3a0b, 0x5c);          // B60 step
```

```
//write_i2c(0x3a0e, 0x06);          // 50Hz max band
//write_i2c(0x3a0d, 0x08);          // 60Hz max band
write_i2c(0x3a14, 0x02);        // 50Hz max exposure
write_i2c(0x3a15, 0xe0);        // 50Hz max exposure
write_i2c(0x4004, 0x02);        // BLC line number
write_i2c(0x3002, 0x1c);        // reset JFIFO, SFIFO, JPG
write_i2c(0x3006, 0xc3);        // disable clock of JPEG2x, JPEG
write_i2c(0x4713, 0x03);        // JPEG mode 3
write_i2c(0x4407, 0x04);        // Quantization scale
write_i2c(0x460b, 0x37);
write_i2c(0x460c, 0x20);
write_i2c(0x4837, 0x16);        // MIPI global timing
write_i2c(0x3824, 0x04);        // PCLK manual divider
write_i2c(0x5001, 0x83);        // SDE on, CMX on, AWB on
write_i2c(0x3503, 0x00);        // AEC/AGC on
```

720p 30fps

```
// YUV 1280x720 30fps
// Input Clock = 24Mhz, PCLK = 84MHz
//same settings as 1280x720 15fps, except the following settings
write_i2c(0x3035, 0x21);        // PLL
```

4.1.6 5M YCbCr Capture

YUV 5M 3.75fps

```
// YUV 2592x1944, 3.75fps
// input clock 24Mhz, PCLK 42Mhz
write_i2c(0x3035, 0x41);        // PLL
write_i2c(0x3036, 0x69);        // PLL
write_i2c(0x3c07, 0x07);        // lightm eter 1 threshold[7:0]
write_i2c(0x3820, 0x40);        // flip
write_i2c(0x3821, 0x06);        // mirror
write_i2c(0x3814, 0x11);        // timing X inc
write_i2c(0x3815, 0x11);        // timing Y inc
write_i2c(0x3800, 0x00);        // HS
write_i2c(0x3801, 0x00);        // HS
write_i2c(0x3802, 0x00);        // VS
write_i2c(0x3803, 0x00);        // VS
write_i2c(0x3804, 0x0a);        // HW (HE)
write_i2c(0x3805, 0x3f);        // HW (HE)
write_i2c(0x3806, 0x07);        // VH (VE)
write_i2c(0x3807, 0x9f);        // VH (VE)
write_i2c(0x3808, 0x0a);        // DVPHO
write_i2c(0x3809, 0x20);        // DVPHO
write_i2c(0x380a, 0x07);        // DVPVO
write_i2c(0x380b, 0x98);        // DVPVO
write_i2c(0x380c, 0x0b);        // HTS
write_i2c(0x380d, 0x1c);        // HTS
```

```
write_i2c(0x380e, 0x07);        // VTS
write_i2c(0x380f, 0xb0);        // VTS
write_i2c(0x3813, 0x04);        // timing V offset
write_i2c(0x3618, 0x04);
write_i2c(0x3612, 0x2b);
write_i2c(0x3709, 0x12);
write_i2c(0x370c, 0x00);
// banding filters are calculated automatically in camera driver
//write_i2c(0x3a02, 0x07);          // 60Hz max exposure
//write_i2c(0x3a03, 0xae);          // 60Hz max exposure
//write_i2c(0x3a08, 0x01);          // B50 step
//write_i2c(0x3a09, 0x27);          // B50 step
//write_i2c(0x3a0a, 0x00);          // B60 step
//write_i2c(0x3a0b, 0xf6);          // B60 step
//write_i2c(0x3a0e, 0x06);          // 50Hz max band
//write_i2c(0x3a0d, 0x08);          // 60Hz max band
//write_i2c(0x3a14, 0x07);          // 50Hz max exposure
//write_i2c(0x3a15, 0xae);          // 50Hz max exposure
write_i2c(0x4004, 0x06);        // BLC line number
write_i2c(0x3002, 0x1c);        // reset JFIFO, SFIFO, JPG
write_i2c(0x3006, 0xc3);        // disable clock of JPEG2x, JPEG
write_i2c(0x4713, 0x02);        // JPEG mode 3
write_i2c(0x4407, 0x0c);        // Quantization sacle
write_i2c(0x460b, 0x37);
write_i2c(0x460c, 0x20);
write_i2c(0x4837, 0x2c);        // MIPI global timing
write_i2c(0x3824, 0x01);        // PCLK manual divider
write_i2c(0x5001, 0x83);        // SDE on, CMX on, AWB on
write_i2c(0x3503, 0x03);
```

YUV 5M 5fps

```
// YUV 2592x1944 5fps
// Input Clock = 24Mhz, PCLK = 56MHz
//same settings as 2592x1944 3.75fps, except the following settings
write_i2c(0x3035, 0x21);        // PLL
write_i2c(0x3036, 0x46);        // PLL
```

YUV 5M 7.5fps

```
// YUV 2592x1944 7.5fps
// Input Clock = 24Mhz, PCLK = 84MHz
//same settings as 2592x1944 3.75fps, except the following settings
write_i2c(0x3035, 0x21);        // PLL
write_i2c(0x3036, 0x69);        // PLL
```

4.1.7 5M JPEG Capture

```
// JPEG, 2592x1944 fixed size, 15fps
// Input clock = 24Mhz, PCLK = 42 MHz
```

```
write_i2c(0x3035, 0x11);       // PLL
write_i2c(0x3036, 0x69);       // PLL
write_i2c(0x3c07, 0x07);       // lightmeter 1 threshold[7:0]
write_i2c(0x3820, 0x40);       // flip
write_i2c(0x3821, 0x26);       // mirror
write_i2c(0x3814, 0x11);       // timing X inc
write_i2c(0x3815, 0x11);       // timing Y inc
write_i2c(0x3800, 0x00);       // HS
write_i2c(0x3801, 0x00);       // HS
write_i2c(0x3802, 0x00);       // VS
write_i2c(0x3803, 0x00);       // VS
write_i2c(0x3804, 0x0a);       // HW (HE)
write_i2c(0x3805, 0x3f);       // HW (HE)
write_i2c(0x3806, 0x07);       // VH (VE)
write_i2c(0x3807, 0x9f);       // VH (VE)
write_i2c(0x3808, 0x0a);       // DVPHO
write_i2c(0x3809, 0x20);       // DVPHO
write_i2c(0x380a, 0x07);       // DVPVO
write_i2c(0x380b, 0x98);       // DVPVO
write_i2c(0x380c, 0x0b);       // HTS
write_i2c(0x380d, 0x1c);       // HTS
write_i2c(0x380e, 0x07);       // VTS
write_i2c(0x380f, 0xb0);       // VTS
write_i2c(0x3813, 0x04);       // timing V offset
write_i2c(0x3618, 0x04);
write_i2c(0x3612, 0x2b);
write_i2c(0x3709, 0x12);
write_i2c(0x370c, 0x00);
/// banding filters are calculated automatically in camera driver
/write_i2c(0x3a02, 0x07);      // 60Hz max exposure
//write_i2c(0x3a03, 0xae);          // 60Hz max exposure
//write_i2c(0x3a08, 0x01);          // B50 step
//write_i2c(0x3a09, 0x27);          // B50 step
//write_i2c(0x3a0a, 0x00);          // B60 step
//write_i2c(0x3a0b, 0xf6);          // B60 step
//write_i2c(0x3a0e, 0x06);          // 50Hz max band
//write_i2c(0x3a0d, 0x08);          // 60Hz max band
//write_i2c(0x3a14, 0x07);          // 50Hz max exposure
//write_i2c(0x3a15, 0xae);          // 50Hz max exposure
write_i2c(0x4004, 0x06);       // BLC line number
write_i2c(0x3002, 0x00);       // reset JFIFO, SFIFO, JPG
write_i2c(0x3006, 0xff);       // disable clock of JPEG2x, JPEG
write_i2c(0x4713, 0x02);       // JPEG mode 2
write_i2c(0x4407, 0x04);       // Quantization sacle
write_i2c(0x460b, 0x35);
write_i2c(0x460c, 0x22);
write_i2c(0x4837, 0x16);       // MIPI global timing
write_i2c(0x3824, 0x04);       // PCLK manual divider
```

```
write_i2c(0x5001, 0x83);        // SDE on, CMX on, AWB on
write_i2c(0x3503, 0x03);
```

4.2 Signal Polarity

The signal polarity of OV5640 and baseband chip/ISP should be set to same to get correct image.

Signal	Register bit	Function
VSYNC	0x4740[0]	1 – data valid Vsync high 0 – data valid Vsync low
PCLK	0x4740[5]	1 – data update at falling-edge 0 – data update at rising-edge
HREF	0x4740[1]	0 – data valid HREF high 1 – data valid HREF low

4.3 Drive Capability

Register bit	Function
0x302c[7:6]	00 – 1x 01 – 2x 10 – 3x 11 – 4x

4.4 I/O Control

Signal	Input/High-z	Output data path	Output - 0	Output - 1
D[9:0]	0x3017[3:0]=0x0 0x3018[7:2]=0x0	0x3017[3:0]=0xf 0x3018[7:2]=0x3f 0x301d[3:0]=0x0 0x301e[7:2]=0x0	0x3017[3:0]=0xf 0x3018[7:2]=0x3f 0x301d[3:0]=0xf 0x301e[7:2]=0x3f 0x301a[3:0]=0x0 0x301b[7:2]=0x0	0x3017[3:0]=0xf 0x3018[7:2]=0x3f 0x301d[3:0]=0xf 0x301e[7:2]=0x3f 0x301a[3:0]=0xf 0x301b[7:2]=0x3f
Vsync	0x3017[6]=0	0x3017[6]=1 0x301d[6]=0	0x3017[6]=1 0x301d[6]=1 0x301a[6]=0	0x3017[6]=1 0x301d[6]=1 0x301a[6]=1
Href	0x3017[5]=0	0x3017[5]=1 0x301d[5]=0	0x3017[5]=1 0x301d[5]=1 0x301a[5]=0	0x3017[5]=1 0x301d[5]=1 0x301a[5]=1
PCLK	0x3017[4]=0	0x3017[4]=1 0x301d[4]=0	0x3017[4]=1 0x301d[4]=1 0x301a[4]=0	0x3017[4]=1 0x301d[4]=1 0x301a[4]=1
Strobe	0x3016[1]=0	0x3016[1]=1 0x301c[1]=0	0x3016[1]=1 0x301c[1]=1 0x3019[1]=0	0x3016[1]=1 0x301c[1]=1 0x3019[1]=1
FREX	0x3017[7]=0	0x3017[7]=1	0x3017[7]=1	0x3017[7]=1

		0x301d[7]=0	0x301d[7]=1 0x301a[7]=0	0x301d[7]=1 0x301a[7]=1
GPIO1	0x3018[1]=0	0x3018[1]=1 0x301e[1]=0	0x3018[1]=1 0x301e[1]=1 0x301b[1]= 0	0x3018[1]=1 0x301e[1]=1 0x301b[1]= 1
GPIO0	0x3018[0]=0	0x3018[0]=1 0x301e[0]=0	0x3018[0]=1 0x301e[0]=1 0x301b[0]= 0	0x3018[0]=1 0x301e[0]=1 0x301b[0]= 1

Note:

1. If any pin is open in camera module or open in system, the pin should be configured as output.

2. If any pin shares same bus with other devices, when OV5640 is set to standby mode and other device is driving the bus, the pin should be set to input/High-z. If any pin shares same bus with other devices, when both OV5640 and other devices are set to standby mode, one pin should set to output to drive the bus and other pin should set to input/high-z.

4.5 YUV Sequence

The YUV sequence of OV5640 output image should match with baseband chip/ISP to get correct image.

0x4300[0:1] control YUV sequence
//Y U Y V
write_i2c(0x4300, 0x30);

//Y V Y U
write_i2c(0x4300, 0x31);

//V Y U Y
write_i2c(0x4300, 0x33);

//U Y V Y
write_i2c(0x4300, 0x32);

4.6 Mirror and Flip

Since OV5640 is a BSI sensor, the light go to sensor from backside of sensor. So the image without mirror/flip looks mirror. The image with mirror looks normal.

i2c_salve_Address = 0x78;

MIRROR (default)

```
reg3820 = read_i2c(0x3820);
reg3820 = reg3820 & 0xf9;     // flip off
write_i2c(0x3820, reg3820);
reg3821 = read_i2c(0x3821);
```

MIRROR

```
reg3821 = reg3821 | 0x06;      // mirror on
write_i2c(0x3821, reg3821);
```

FLIP

```
reg3820 = read_i2c(0x3820);
reg3820 = reg3820 | 0x06;      // flip on
write_i2c(0x3820, reg3820);
reg3821 = read_i2c(0x3821);
reg3821 = reg3821 & 0xf9;      // mirror off
write_i2c(0x3821, reg3821);
```

FLIP

MIRROR&FLIP

```
reg3820 = read_i2c(0x3820);
reg3820 = reg3820 | 0x06;      // flip on
write_i2c(0x3820, reg3820);
reg3821 = read_i2c(0x3821);
reg3821 = reg3821 | 0x06;      // mirror on
write_i2c(0x3821, reg3821)
```

MIRROR&FLIP

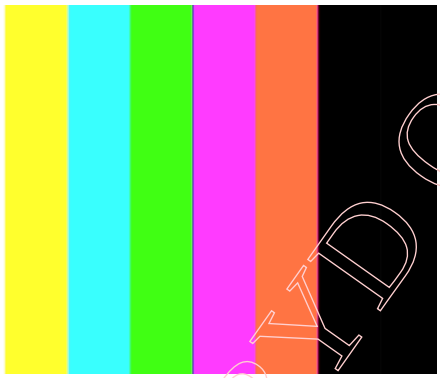
Normal

```
reg3820 = read_i2c(0x3820);
reg3820 = reg3820 & 0xf9;      // flip off
write_i2c(0x3820, reg3820);
reg3821 = read_i2c(0x3821);
reg3821 = reg3821 & 0xf9;      // mirror off
write_i2c(0x3821, reg3821);
```

NORML

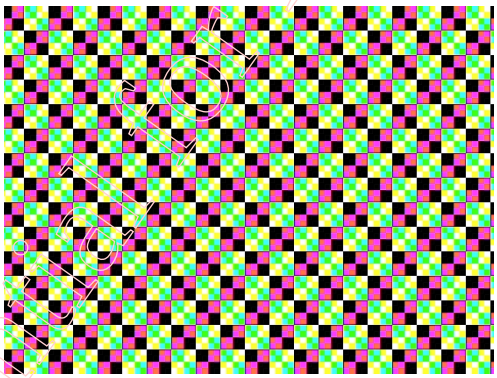
4.7 Test Pattern

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



Color bar

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



Color square

4.8 Remove Light Band

OV5640_set_bandingfilter() function set banding filters automatically in camera driver.

Light band is removed by set exposure to $\frac{n}{100}$ ($\frac{n}{120}$ for 60Hz) seconds. The banding filter value tell OV5640 how many lines is $\frac{n}{100}$ ($\frac{n}{120}$ for 60Hz) seconds.

$$Banding\_filter\_50hz = \frac{\frac{1}{100}}{line\_period} = \frac{\frac{1}{100}}{HTS \times sysclk} = \frac{sysclk}{HTS \times 100}$$

$$Banding\_filter\_60hz = \frac{\frac{1}{120}}{line\_period} = \frac{\frac{1}{120}}{HTS \times sysclk} = \frac{sysclk}{HTS \times 120}$$

HTS = register {0x380c, 0x380d}

sysclk could be calculated by the function OV5640_get_sysclk().

4.9 User Interface Functions

4.9.1 Brightness

Brightness +4
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x40);
write_i2c(0x5588, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Brightness +3
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x30);
write_i2c(0x5588, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Brightness +2
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x20);
write_i2c(0x5588, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

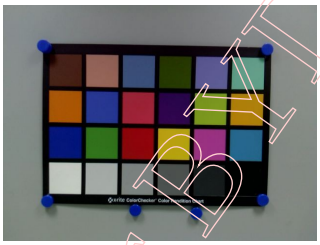
Brightness +1
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x10);
write_i2c(0x5588, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Default Brightness
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x00);
write_i2c(0x5588, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Brightness -1
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x10);
write_i2c(0x5588, 0x09);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Brightness -2
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x20);
write_i2c(0x5588, 0x09);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Brightness -3
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x30);
write_i2c(0x5588, 0x09);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3



Brightness -4
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5587, 0x40);
write_i2c(0x5588, 0x09);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

4.9.2 Contrast

Contrast +3
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5586, 0x2c);
write_i2c(0x5585, 0x1c);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Contrast +2
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5586, 0x28);
write_i2c(0x5585, 0x18);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Contrast +1
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5586, 0x24);
write_i2c(0x5585, 0x10);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Contrast Standard
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5586, 0x20);
write_i2c(0x5585, 0x00);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Contrast -1
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5586, 0x1c);
write_i2c(0x5585, 0x1c);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Contrast -2
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5586, 0x18);
write_i2c(0x5585, 0x18);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Contrast -3
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5586, 0x14);
write_i2c(0x5585, 0x14);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

4.9.3 Saturation

Saturation +3
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5381, 0x1c);
write_i2c(0x5382, 0x5a);
write_i2c(0x5383, 0x06);
write_i2c(0x5384, 0x2b);
write_i2c(0x5385, 0xab);
write_i2c(0x5386, 0xd6);
write_i2c(0x5387, 0xda);
write_i2c(0x5388, 0xd6);
write_i2c(0x5389, 0x04);
write_i2c(0x538b, 0x98);
write_i2c(0x538a, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Saturation +2

```
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5381, 0x1c);
write_i2c(0x5382, 0x5a);
write_i2c(0x5383, 0x06);
write_i2c(0x5384, 0x24);
write_i2c(0x5385, 0x8f);
write_i2c(0x5386, 0xb3);
write_i2c(0x5387, 0xb6);
write_i2c(0x5388, 0xb3);
write_i2c(0x5389, 0x03);
write_i2c(0x538b, 0x98);
write_i2c(0x538a, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3
```

Saturation +1

```
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5381, 0x1c);
write_i2c(0x5382, 0x5a);
write_i2c(0x5383, 0x06);
write_i2c(0x5384, 0x1f);
write_i2c(0x5385, 0x7a);
write_i2c(0x5386, 0x9a);
write_i2c(0x5387, 0x9c);
write_i2c(0x5388, 0x9a);
write_i2c(0x5389, 0x02);
write_i2c(0x538b, 0x98);
write_i2c(0x538a, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3
```

Saturation Standard

```
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5381, 0x1c);
write_i2c(0x5382, 0x5a);
write_i2c(0x5383, 0x06);
write_i2c(0x5384, 0x1a);
write_i2c(0x5385, 0x66);
write_i2c(0x5386, 0x80);
write_i2c(0x5387, 0x82);
write_i2c(0x5388, 0x80);
write_i2c(0x5389, 0x02);
write_i2c(0x538b, 0x98);
write_i2c(0x538a, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3
```

Saturation -1

```
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5381, 0x1c);
write_i2c(0x5382, 0x5a);
write_i2c(0x5383, 0x06);
write_i2c(0x5384, 0x15);
write_i2c(0x5385, 0x52);
write_i2c(0x5386, 0x66);
write_i2c(0x5387, 0x68);
write_i2c(0x5388, 0x66);
write_i2c(0x5389, 0x02);
write_i2c(0x538b, 0x98);
write_i2c(0x538a, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3
```

Saturation -2

```
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5381, 0x1c);
write_i2c(0x5382, 0x5a);
write_i2c(0x5383, 0x06);
write_i2c(0x5384, 0x10);
write_i2c(0x5385, 0x3d);
write_i2c(0x5386, 0x4d);
write_i2c(0x5387, 0x4e);
write_i2c(0x5388, 0x4d);
write_i2c(0x5389, 0x01);
write_i2c(0x538b, 0x98);
write_i2c(0x538a, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3
```

Saturation -3

```
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5381, 0x1c);
write_i2c(0x5382, 0x5a);
write_i2c(0x5383, 0x06);
write_i2c(0x5384, 0x0c);
write_i2c(0x5385, 0x30);
write_i2c(0x5386, 0x3d);
write_i2c(0x5387, 0x3e);
write_i2c(0x5388, 0x3d);
write_i2c(0x5389, 0x01);
write_i2c(0x538b, 0x98);
write_i2c(0x538a, 0x01);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3
```

4.9.4 EV

EV +3
write_i2c(0x3a0f, 0x60);
write_i2c(0x3a10, 0x58);
write_i2c(0x3a11, 0xa0);
write_i2c(0x3a1b, 0x60);
write_i2c(0x3a1e, 0x58);
write_i2c(0x3a1f, 0x20);

EV +2
write_i2c(0x3a0f, 0x50);
write_i2c(0x3a10, 0x48);
write_i2c(0x3a11, 0x90);
write_i2c(0x3a1b, 0x50);
write_i2c(0x3a1e, 0x48);
write_i2c(0x3a1f, 0x20);

EV +1
write_i2c(0x3a0f, 0x40);
write_i2c(0x3a10, 0x38);
write_i2c(0x3a11, 0x71);
write_i2c(0x3a1b, 0x40);
write_i2c(0x3a1e, 0x38);
write_i2c(0x3a1f, 0x10);

EV Standard
write_i2c(0x3a0f, 0x38);
write_i2c(0x3a10, 0x30);
write_i2c(0x3a11, 0x61);
write_i2c(0x3a1b, 0x38);
write_i2c(0x3a1e, 0x30);
write_i2c(0x3a1f, 0x10);

EV -1
write_i2c(0x3a0f, 0x30);
write_i2c(0x3a10, 0x28);
write_i2c(0x3a11, 0x61);
write_i2c(0x3a1b, 0x30);
write_i2c(0x3a1e, 0x28);
write_i2c(0x3a1f, 0x10);

EV -2
write_i2c(0x3a0f, 0x20);
write_i2c(0x3a10, 0x18);
write_i2c(0x3a11, 0x41);
write_i2c(0x3a1b, 0x20);
write_i2c(0x3a1e, 0x18);
write_i2c(0x3a1f, 0x10);

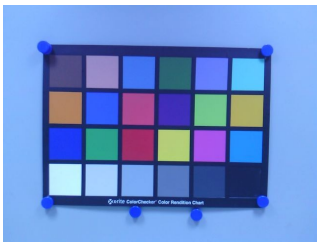
EV -3
write_i2c(0x3a0f, 0x10);
write_i2c(0x3a10, 0x08);
write_i2c(0x3a1b, 0x10);
write_i2c(0x3a1e, 0x08);
write_i2c(0x3a11, 0x20);
write_i2c(0x3a1f, 0x10);

4.9.5 Light Mode

Auto
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x3406, 0x00);
write_i2c(0x3400, 0x04);
write_i2c(0x3401, 0x00);
write_i2c(0x3402, 0x04);
write_i2c(0x3403, 0x00);
write_i2c(0x3404, 0x04);
write_i2c(0x3405, 0x00);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // lanuch group 3

Sunny
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x3406, 0x01);
write_i2c(0x3400, 0x06);
write_i2c(0x3401, 0x1c);
write_i2c(0x3402, 0x04);
write_i2c(0x3403, 0x00);
write_i2c(0x3404, 0x04);
write_i2c(0x3405, 0xf3);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // lanuch group 3

Office
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x3406, 0x01);
write_i2c(0x3400, 0x05);
write_i2c(0x3401, 0x48);
write_i2c(0x3402, 0x04);
write_i2c(0x3403, 0x00);
write_i2c(0x3404, 0x07);
write_i2c(0x3405, 0xcf);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // lanuch group 3



Cloudy

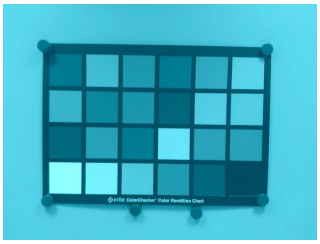
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x3406, 0x01);
write_i2c(0x3400, 0x06);
write_i2c(0x3401, 0x48);
write_i2c(0x3402, 0x04);
write_i2c(0x3403, 0x00);
write_i2c(0x3404, 0x04);
write_i2c(0x3405, 0xd3);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // lanuch group 3

Home
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x3406, 0x01);
write_i2c(0x3400, 0x04);
write_i2c(0x3401, 0x10);
write_i2c(0x3402, 0x04);
write_i2c(0x3403, 0x00);
write_i2c(0x3404, 0x08);
write_i2c(0x3405, 0x40);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // lanuch group 3

4.9.6 Special Effects

Normal (off)
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x06);
write_i2c(0x5583, 0x40); // sat U
write_i2c(0x5584, 0x10); // sat V
write_i2c(0x5003, 0x08);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

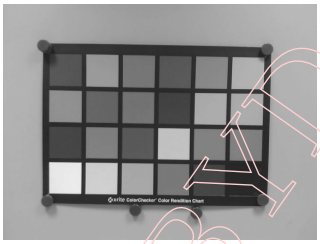
Blueish (cool light)
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x1e);
write_i2c(0x5583, 0xa0);
write_i2c(0x5584, 0x40);
write_i2c(0x5003, 0x08);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3



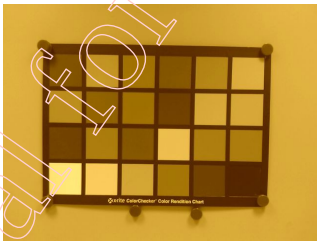
Redish (warm)
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x1e);
write_i2c(0x5583, 0x80);
write_i2c(0x5584, 0xc0);

write_i2c(0x5003, 0x08);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

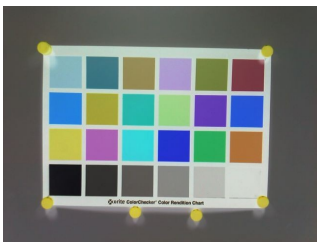
Black and white
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x1e);
write_i2c(0x5583, 0x80);
write_i2c(0x5584, 0x80);
write_i2c(0x5003, 0x08);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3



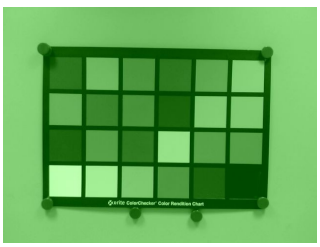
Sepia
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x1e);
write_i2c(0x5583, 0x40);
write_i2c(0x5584, 0xa0);
write_i2c(0x5003, 0x08);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3



Negative
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x40);
write_i2c(0x5003, 0x08);
write_i2c(0x5583, 0x40); // sat U
write_i2c(0x5584, 0x10); // sat V
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3



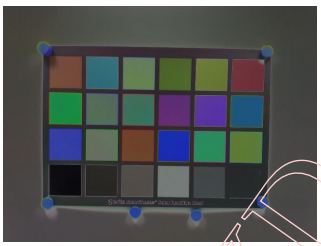
Greenish
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x1e);
write_i2c(0x5583, 0x60);
write_i2c(0x5584, 0x60);
write_i2c(0x5003, 0x08);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3



Overexposure
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x1e);
write_i2c(0x5583, 0xf0);
write_i2c(0x5584, 0xf0);
write_i2c(0x5003, 0x08);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3

Solarize

```
write_i2c(0x3212, 0x03); // start group 3
write_i2c(0x5580, 0x06);
write_i2c(0x5583, 0x40); // sat U
write_i2c(0x5584, 0x10); // sat V
write_i2c(0x5003, 0x09);
write_i2c(0x3212, 0x13); // end group 3
write_i2c(0x3212, 0xa3); // launch group 3
```



4.9.7 Night Mode

When night mode is turned on, dummy lines is inserted automatically and the frame rate is decreased. The total dummy line inserted in controlled by register 0x3a02, 0x3a03 for 50hz light and 0x3a14, 0x3a15 for 60hz light. The night mode on/off switch is controlled by register 0x3a00 bit[2].

Night Mode On

```
temp = read_i2c(0x3a00);
temp = temp | 0x04;
write_i2c(0x3a00, temp);
```

Night Mode Off

```
temp = read_i2c(0x3a00);
temp = temp & 0xfb;
write_i2c(0x3a00, temp);
```

4.7.8 Banding Filter Selection

The camera driver of OV5640 supports banding filter calculation. So there is no need to set banding filter manually. There are 4 options for banding filter, they are

Off

```
temp = read_i2c(0x3a00);
temp = temp & 0xdf;       // turn off banding filter
write_i2c(0x3a00, temp);
```

Manual 50Hz

```
write_i2c(0x3c00, 04);    // set to 50Hz
write_i2c(0x3c01, 80);    // manual banding filter
temp = read_i2c(0x3a00);
temp = temp | 0x20;       // turn on banding filter
write_i2c(0x3a00, temp);
```

Manual 60Hz

```
write_i2c(0x3c00, 00);    // set to 60Hz
write_i2c(0x3c01, 80);    // manual banding filter
temp = read_i2c(0x3a00);
```

```
temp = temp | 0x20;        // turn on banding filter
write_i2c(0x3a00, temp);
```

Auto Detection

```
write_i2c(0x3c01, 00);     // auto banding filter
temp = read_i2c(0x3a00);
temp = temp & 0xdf;        // turn off banding filter
write_i2c(0x3a00, temp);
```

The auto detection function depends on input clock. If input clock is not 24Mhz, please contact with OmniVision local FAE for setting update.

4.10 Auto Focus

4.10.1 Embedded Auto Focus Solution

The auto focus function of OV5640 is controlled by built-in micro-controller, and the AF driver is also built in OV5640 sensor. The firmware of micro-controller is downloaded from host. After firmware is running, the built-in micro-controller of OV5647 read auto focus information from sensor, calculate the focus position and driver VCM to the position. The host controls the function of built-in micro-controller by I2C commends.

4.10.2 I2C Commands for Auto Focus

Register name	Address	Description	Value
CMD_MAIN	0x3022	Auto focus main commend register	0x03 - Trig Single Auto Focus 0x06 - Pause Auto Focus 0x08 - Release Focus 0x12 - Re-launch Focus Zones 0x00 – command is finished
CMD_ACK	0x3023	ACK of commend	0x00 - Commend is finished 0x01 - Commend is running
FW_STATUS	0x3029	Status of focus	0x7F - S_FIRWARE Firmware is downloaded and not run. Cause 1 MCU is off Cause 2 Firmware is not correct. 0x7E - S_STARTUP Firmware is initializing. 0x70 - S_IDLEIdle state, focus is released. lens is located at the furthest position. 0x00 - S_FOCUSING Auto Focus is running. 0x10 S_FOCUSED Auto Focus is completed.

Note: The MCU will auto clear CMD_MAIN to zero after the command is receipt, and auto clear CMD_ACK to zero when the command is completed.

4.10.3 AF Sequence

There is registers read and written by using OV5640 auto focus firmware sequence,

The auto focus steps as below:

1. Download firmware

   when enter preview for the first time

2. Auto focus

   when capture

3. Release focus

   when finished capture and return to prreview

4.10.4 Download firmware

After initial OV5640, the AF firmware can be download. It is same as download the initial setting. To speedup firmware download, I2c multiple bytes write is highly recommended.

After download the firmware, please check the registers below:

MCU on:0x3000 BIT6=0 BIT5=0 0x3004 BIT6=1 BIT5=1

AFC on :0x3001 BIT6=00x3005 BIT6=1

4.10.5 Auto focus

Auto focus should be completed before capture sequence started.

1. Write 0x03 to register 0x3022 to start single focus
2. read back register 0x3029, if the read back value is 0x10, then focus is finished.
3. Write 0x06 to register 0x3022 to pause auto focus. Lens will stay in current focus position.

4.10.6 Release Focus

After capture , Write 0x08 to register 0x3022 to release the position of lens, let the lens move back infinite.

4.11 Capture Sequence

4.11.1 Shutter

The shutter of OV5640 controls exposure time. The unit of shutter is line period. Shutter value has limitation for each resolution. The maximum shutter value is VTS(0x380e, 0x380f) - 4.

Shutter register value is 16 multiple exposure lines. The shutter value are stored in 3 registers, reg0x3500, reg0x3501,reg3502 .

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

4.11.2 Gain

Gain is stored in reg0x350a and Reg0x350b.

Gain = ((reg0x350a & 03)<<4) + (reg0x350b>>4);

4.11.3 Dummy Lines and Dummy Pixels

If enable auto-night mode of OV5640, the dummy lines are automatic inserted. The number of dummy lines are controled by register {0x3a02[3:0], 0x3a03} for 60Hz and {0x3a14[3:0], 0x3a15} for 50Hz.

Dummy pixels are used to adjust timing. Please adjust HTS register {0x380c, 0x380d} to insert dummy pixels.

4.11.4 Capture Sequence

4.11.4.1 Auto Focus

Trigger single auto focus

Pause auto focus (hold lens at focused position)

4.11.4.2 Read Preview Registers

Read preview_shutter
Read preview_gain
Read preview_HTS
Read preview_sysclk
Read preview_bining_factor B

4.11.4.3 Change Resolution to Capture

Download Capture setting. Turn off AGC/AEC.

4.11.4.4 Read Capture Register Values

Read capture _HTS
Read capture _VTS
Read capture_banding_filter
Read capture SYSCLK

4.11.4.5 Preview Gain/Exposure to Capture/Gain Exposure

The shutter of capture should have same time as preview.

$$capture\_shutter = preview\_shutter \times \frac{capture\_PCLK}{preview\_PCLK} \times \frac{preview\_HTS}{capture\_HTS} \times B$$

$$capture\_gain = preview\_gain$$

Where B is binning factor. B=1 for OV5640.

Under strong light, when preview_shutter = 1, after calculation capture_shutter=0 due to error of integer calculation. To avoid such issue, we may calculate like this

$$preview\_shutter = capture\_shutter$$

$$capture\_gain = preview\_gain \times \frac{capture\_PCLK}{preview\_PCLK} \times \frac{preview\_HTS}{capture\_HTS} \times B$$

4.11.4.6 Gain to Exposure and Capture Banding Filter

```
gain_exposure = capture_shutter*capture_gain;

If ( gain_exposure < capture_banding_filter) {
  // capture_shutter < 1/100
  capture_shutter = gain_exposure;
  capture_gain = gain_exposure/capture_shutter;
}
else {
  if ( gain_exposure > (capture_VTS -4)) {
        // exposure reach maximum
        capture_shutter = capture_VTS -4;
        capture_gain = gain_exposure/capture_shutter;
  }
  else {
  // 1/100 < exposure < max, capture_shutter = N * capture_banding_filter
  capture_shutter = int (gain_exposure/capture_banding_filter) * capture_banding_filter;
  capture_gain = gain_exposure/capture_shutter;
  }
}
```

4.11.4.7 Write gain/exposure value

Write capture_gain
Write capture_shutter

4.11.4.8 Capture

Wait for 2 Vsync
Capture the 3rd frame.

4.11.4.9 Back to Preview

// release auto focus, release lens back to infinite.

//Write Registers, Change to preview

//Start AG/AE

write_i2c(0x3503, 0);

4.12 Scale and Zoom

To use scale/zoom function, the ISP scale 0x5001[5] must be set to '1'. To make sure zoom/scale changes smoothly, I2c group write is highly recommended.

4.12.1 Scale

OV5640 ISP support down scale. Images could be scaled down to any size less than 2592x1944. If the aspect ratio of output image is same as sensor image (4:3), adjust register DVPHO and DVPVO to get the desired image size. For example, to get 2048x1536 output

DVPHO = 2048

DVPVO = 1536

H offset = 16

Y offset = 4

```
write_i2c(0x3212, 0x03);      // start group 3
write_i2c(0x3808, 0x08);      // DVPHO = 2048
write_i2c(0x3809, 0x00);      // DVP HO
write_i2c(0x380a, 0x06);      // DVPVO = 1536
write_i2c(0x380b, 0x00);      // DVPVO
write_i2c(0x3810, 0x00);      // H offset = 16
write_i2c(0x3811, 0x10);      // H offset
write_i2c(0x3812, 0x00);      // V offset = 4
write_i2c(0x3813, 0x04);      // V offset
write_i2c(0x3212, 0x13);      // end group 3
write_i2c(0x3212, 0xa3);      // launch group 3
```

If the aspect ratio of output image is different as sensor image (not 4:3), then need to adjust X offset and Y offset to clip input image to same aspect ratio as output image. For example, to get 1280x1024 output

DVPHO = 1280

DVPVO = 1024

input image height = 1944

input image width = 1944 * 1280/1024 = 2430

X offset = 16

Y offset = 4 + (2592 - 2430)/2 = 85

```
write_i2c(0x3212, 0x03);        // start group 3
write_i2c(0x3808, 0x05);        // DVPHO = 1280
write_i2c(0x3809, 0x00);        // DVP HO
write_i2c(0x380a, 0x04);        // DVPVO = 1024
write_i2c(0x380b, 0x00);        // DVPVO
write_i2c(0x3810, 0x00);        // H offset = 16
write_i2c(0x3811, 0x10);        // H offset
write_i2c(0x3812, 0x00);        // V offset = 85
write_i2c(0x3813, 0x55);        // V offset
write_i2c(0x3212, 0x13);        // end group 3
write_i2c(0x3212, 0xa3);        // launch group 3
```

4.12.2 Digital Zoom

For any image output size less than 2592x1944, increase X offset and Y offset in such way that the aspect ratio of input image match with aspect ratio of output image, digital zoom function is implemented by this way. For example, when output image is 1600x1200

Digital Zoom 1x

DVPHO = 1600

DVPVO = 1200

X offset = 16

Y offset = 4

```
write_i2c(0x3212, 0x03);        // start group 3
write_i2c(0x3808, 0x06);        // DVPHO = 1600
write_i2c(0x3809, 0x40);        // DVP HO
write_i2c(0x380a, 0x04);        // DVPVO = 1200
write_i2c(0x380b, 0xb0);        // DVPVO
write_i2c(0x3810, 0x00);        // H offset = 16
write_i2c(0x3811, 0x10);        // H offset
write_i2c(0x3812, 0x00);        // V offset = 4
write_i2c(0x3813, 0x04);        // V offset
write_i2c(0x3212, 0x13);        // end group 3
write_i2c(0x3212, 0xa3);        // launch group 3
```

Digital Zoom 1.5x

DVPHO = 1600

DVPVO = 1200

input image width = 2592/1.5 = 1728 > 1600

input image height = 1944/1.5 = 1296 > 1200

X offset = 16 + (2592 – 1728)/2 = 448

Y offset = 4 + (1944 – 1296)/2 = 328

```
write_i2c(0x3212, 0x03);        // start group 3
write_i2c(0x3808, 0x06);        // DVPHO = 1600
write_i2c(0x3809, 0x40);        // DVP HO
write_i2c(0x380a, 0x04);        // DVPVO = 1200
write_i2c(0x380b, 0xb0);        // DVPVO
write_i2c(0x3810, 0x01);        // H offset = 448
write_i2c(0x3811, 0xc0);        // H offset
write_i2c(0x3812, 0x01);        // V offset = 328
write_i2c(0x3813, 0x48);        // V offset
write_i2c(0x3212, 0x13);        // end group 3
write_i2c(0x3212, 0xa3);        // launch group 3
```

Appendix I Sample Code of Camera Driver

```
int m_iCombo_NightMode = 0;

int XVCLK = 2400;     // real clock/10000
int preview_sysclk, preview_HTS, preview_VTS;
int AE_Target = 52;
int AE_high, AE_low;

int OV5640_init_setting()
{
        // initialize OV5640
        int regInit[] =
        {
                //
                //OV5640 setting Version History
                //dated 04/08/2010 A02
                //--Based on v08 release
                //
                //dated 04/20/2010 A03
                //--Based on V10 release
                //
                //dated 04/22/2010 A04
                //--Based on V10 release
                //--updated ccr & awb setting
                //
                //dated 04/22/2010 A06
                //--Based on A05 release
                //--Add pg setting
                //
                //dated 05/19/2011 A09
                //--changed pchg 3708 setting

                0x3008, 0x42,     // software power down
                0x3103, 0x03,     // sysclk from pll
                0x3017, 0xff,     // Frex, Vsync, Href, PCLK, D[9:6] output
                0x3018, 0xff,     // D[5:0], GPIO[1:0] output
                0x3034, 0x1a,     // PLL, MIPI 10-bit
                0x3037, 0x13,     // PLL
                0x3108, 0x01,     // clock divider
                0x3630, 0x36,
                0x3631, 0x0e,
                0x3632, 0xe2,
                0x3633, 0x12,
                0x3621, 0xe0,
                0x3704, 0xa0,
                0x3703, 0x5a,
                0x3715, 0x78,
                0x3717, 0x01,
                0x370b, 0x60,
                0x3705, 0x1a,
                0x3905, 0x02,
                0x3906, 0x10,
                0x3901, 0x0a,
                0x3731, 0x12,
                0x3600, 0x08,     // VCM debug
                0x3601, 0x33,     // VCM debug
                0x302d, 0x60,     // system control
                0x3620, 0x52,
                0x371b, 0x20,
                0x471c, 0x50,
```

```
0x3a13, 0x43,    // pre-gain = 1.05x
0x3a18, 0x00,    // AEC gain ceiling = 7.75x
0x3a19, 0x7c,    // AEC gain ceiling
0x3635, 0x13,
0x3636, 0x03,
0x3634, 0x40,
0x3622, 0x01,

//50/60Hz detection
0x3c01, 0x34,    // sum auto, band counter enable, threshold = 4
0x3c04, 0x28,    // threshold low sum
0x3c05, 0x98,    // threshold high sum
0x3c06, 0x00,    // light meter 1 threshold H
0x3c07, 0x07,    // light meter 1 threshold L
0x3c08, 0x00,    // light meter 2 threshold H
0x3c09, 0x1c,    // light meter 2 threshold L
0x3c0a, 0x9c,    // sample number H
0x3c0b, 0x40,    // sample number L
0x3810, 0x00,    // X offset
0x3811, 0x10,    // X offset
0x3812, 0x00,    // Y offset
0x3708, 0x64,
0x4001, 0x02,    // BLC start line
0x4005, 0x1a,    // BLC always update
0x3000, 0x00,    // enable MCU, OTP
0x3004, 0xff,    // enable BIST, MCU memory, MCU, OTP, STROBE, D5060, timing, array clock
0x300e, 0x58,    // MIPI 2 lane? power down PHY HS TX, PHY LP RX, DVP enable
0x302e, 0x00,
0x4300, 0x30,    // YUV 422, YUYV
0x501f, 0x00,    // ISP YUV 422
0x440e, 0x00,
0x5000, 0xa7,    // LENC on, raw gamma on, BPC on, WPC on, CIP on

//AEC target
0x3a0f, 0x30,    // stable in high
0x3a10, 0x28,    // stable in low
0x3a1b, 0x30,    // stable out high
0x3a1e, 0x26,    // stable out low
0x3a11, 0x60,    // fast zone high
0x3a1f, 0x14,    // fast zone low

//LENC
0x5800, 0x23,
0x5801, 0x14,
0x5802, 0x0f,
0x5803, 0x0f,
0x5804, 0x12,
0x5805, 0x26,
0x5806, 0x0c,
0x5807, 0x08,
0x5808, 0x05,
0x5809, 0x05,
0x580a, 0x08,
0x580b, 0x0d,
0x580c, 0x08,
0x580d, 0x03,
0x580e, 0x00,
0x580f, 0x00,
0x5810, 0x03,
0x5811, 0x09,
0x5812, 0x07,
0x5813, 0x03,
0x5814, 0x00,
0x5815, 0x01,
```

```
0x5816, 0x03,
0x5817, 0x08,
0x5818, 0x0d,
0x5819, 0x08,
0x581a, 0x05,
0x581b, 0x06,
0x581c, 0x08,
0x581d, 0x0e,
0x581e, 0x29,
0x581f, 0x17,
0x5820, 0x11,
0x5821, 0x11,
0x5822, 0x15,
0x5823, 0x28,
0x5824, 0x46,
0x5825, 0x26,
0x5826, 0x08,
0x5827, 0x26,
0x5828, 0x64,
0x5829, 0x26,
0x582a, 0x24,
0x582b, 0x22,
0x582c, 0x24,
0x582d, 0x24,
0x582e, 0x06,
0x582f, 0x22,
0x5830, 0x40,
0x5831, 0x42,
0x5832, 0x24,
0x5833, 0x26,
0x5834, 0x24,
0x5835, 0x22,
0x5836, 0x22,
0x5837, 0x26,
0x5838, 0x44,
0x5839, 0x24,
0x583a, 0x26,
0x583b, 0x28,
0x583c, 0x42,
0x583d, 0xce,    // LENC BR offset

//AWB
0x5180, 0xff,    // AWB B block
0x5181, 0xf2,    // AWB control
0x5182, 0x00,    // [7:4] max local counter, [3:0] max fast counter
0x5183, 0x14,    // AWB advance
0x5184, 0x25,
0x5185, 0x24,
0x5186, 0x09,
0x5187, 0x09,
0x5188, 0x09,
0x5189, 0x75,
0x518a, 0x54,
0x518b, 0xe0,
0x518c, 0xb2,
0x518d, 0x42,
0x518e, 0x3d,
0x518f, 0x56,
0x5190, 0x46,
0x5191, 0xf8,    // AWB top limit
0x5192, 0x04,    // AWB botton limit
0x5193, 0x70,    // Red limit
0x5194, 0xf0,    // Green Limit
0x5195, 0xf0,    // Blue limit
```

```
0x5196, 0x03,    // AWB control
0x5197, 0x01,    // local limit
0x5198, 0x04,
0x5199, 0x12,
0x519a, 0x04,
0x519b, 0x00,
0x519c, 0x06,
0x519d, 0x82,
0x519e, 0x38,    // AWB control

//Gamma
0x5480, 0x01,    // BIAS plus on
0x5481, 0x08,
0x5482, 0x14,
0x5483, 0x28,
0x5484, 0x51,
0x5485, 0x65,
0x5486, 0x71,
0x5487, 0x7d,
0x5488, 0x87,
0x5489, 0x91,
0x548a, 0x9a,
0x548b, 0xaa,
0x548c, 0xb8,
0x548d, 0xcd,
0x548e, 0xdd,
0x548f, 0xea,
0x5490, 0x1d,

//color matrix
0x5381, 0x1e,    // CMX1 for Y
0x5382, 0x5b,    // CMX2 for Y
0x5383, 0x08,    // CMX3 for Y
0x5384, 0x0a,    // CMX4 for U
0x5385, 0x7e,    // CMX5 for U
0x5386, 0x88,    // CMX6 for U
0x5387, 0x7c,    // CMX7 for V
0x5388, 0x6c,    // CMX8 for V
0x5389, 0x10,    // CMX9 for V
0x538a, 0x01,    // sign[9]
0x538b, 0x98,    // sign[8:1]

//UV adjust
0x5580, 0x06,    // brightness on, saturation on
0x5583, 0x40,    // Sat U
0x5584, 0x10,    // Sat V
0x5589, 0x10,    // UV adjust th1
0x558a, 0x00,    // UV adjust th2[8]
0x558b, 0xf8,    // UV adjust th2[7:0]
0x501d, 0x40,    // enable manual offset in contrast

//CIP
0x5300, 0x08,    // sharpen-MT th1
0x5301, 0x30,    // sharpen-MT th2
0x5302, 0x10,    // sharpen-MT off1
0x5303, 0x00,    // sharpen-MT off2
0x5304, 0x08,    // De-noise th1
0x5305, 0x30,    // De-noise th2
0x5306, 0x08,    // De-noise off1
0x5307, 0x16,    // De-noise off2
0x5309, 0x08,    // sharpen-TH th1
0x530a, 0x30,    // sharpen-TH th2
0x530b, 0x04,    // sharpen-TH off1
0x530c, 0x06,    // sharpen-TH off2
```

```
                0x5025, 0x00,
                0x3008, 0x02,    // wake up from software power down
        };

        OV5640_write_i2c(0x3103, 0x11);        // sysclk from pad
        OV5640_write_i2c(0x3008, 0x82);        // software reset

        // delay 5ms
        Delay(5);

        // Write initialization table
        for (int i=0; i<sizeof(regInit)/sizeof(int); i+=2)
        {
                OV5640_write_i2c(regInit[i], regInit[i+1]);
        }

        return 0;
}

int OV5640_preview_setting()
{
        // set OV5640 to preview mode
        int regPreview[] =
        {
                // 640x480 15fps, night mode 5fps
                // Input CLock = 24Mhz
                // PCLK = 17Mhz
                0x3035, 0x21,    // PLL
                0x3036, 0x46,    // PLL
                0x3c07, 0x08,    // lightmeter 1 threshold[7:0]
                0x3820, 0x41,    // flip
                0x3821, 0x07,    // mirror
                0x3814, 0x31,    // timing X inc
                0x3815, 0x31,    // timing Y inc
                0x3800, 0x00,    // HS
                0x3801, 0x00,    // HS
                0x3802, 0x00,    // VS
                0x3803, 0x04,    // VS
                0x3804, 0x0a,    // HW (HE)
                0x3805, 0x3f,    // HW (HE)
                0x3806, 0x07,    // VH (VE)
                0x3807, 0x9b,    // VH (VE)
                0x3808, 0x02,    // DVPHO
                0x3809, 0x80,    // DVPHO
                0x380a, 0x01,    // DVPVO
                0x380b, 0xe0,    // DVPVO
                0x380c, 0x07,    // HTS
                0x380d, 0x68,    // HTS
                0x380e, 0x03,    // VTS
                0x380f, 0xd8,    // VTS
                0x3813, 0x06,    // timing V offset
                0x3618, 0x00,
                0x3612, 0x29,
                0x3709, 0x52,
                0x370c, 0x03,
                0x3a02, 0x0b,    // 60Hz max exposure, night mode 5fps
                0x3a03, 0x88,    // 60Hz max exposure
                0x3a14, 0x0b,    // 50Hz max exposure, night mode 5fps
                0x3a15, 0x88,    // 50Hz max exposure
                0x4004, 0x02,    // BLC line number
                0x3002, 0x1c,    // reset JFIFO, SFIFO, JPG
                0x3006, 0xc3,    // disable clock of JPEG2x, JPEG
                0x4713, 0x03,    // JPEG mode 3
```

```
                0x4407, 0x04,    // Quantization sacle
                0x460b, 0x35,
                0x460c, 0x22,
                0x4837, 0x22,    // MIPI global timing
                0x3824, 0x02,    // PCLK manual divider
                0x5001, 0xa3,    // SDE on, CMX on, AWB on
                0x3503, 0x00,     // AEC/AGC on
        };

        // Write preview table
        for (int i=0; i<sizeof(regPreview)/sizeof(int); i+=2)
        {
                OV5640_write_i2c(regPreview[i], regPreview[i+1]);
        }

        return 0;
}

int OV5640_video_setting()
{
        // set OV5640 to video mode
        int regVideo[] =
        {
                // input clock 24Mhz
                // PCLK 42Mhz
                0x3035, 0x41,    // PLL
                0x3036, 0x69,    // PLL
                0x3c07, 0x07,    // lightmeter 1 threshold[7:0]
                0x3820, 0x41,    // flip
                0x3821, 0x07,    // mirror
                0x3814, 0x31,    // timing X inc
                0x3815, 0x31,    // timing Y inc
                0x3800, 0x00,    // HS
                0x3801, 0x00,    // HS
                0x3802, 0x00,    // VS
                0x3803, 0xfa,    // VS
                0x3804, 0x0a,    // HW (HE)
                0x3805, 0x3f,    // HW (HE)
                0x3806, 0x06,    // VH (VE)
                0x3807, 0xa9,    // VH (VE)
                0x3808, 0x05,    // DVPHO
                0x3809, 0x00,    // DVPHO
                0x380a, 0x02,    // DVPVO
                0x380b, 0xd0,    // DVPVO
                0x380c, 0x07,    // HTS
                0x380d, 0x64,    // HTS
                0x380e, 0x02,    // VTS
                0x380f, 0xe4,    // VTS
                0x3813, 0x04,    // timing V offset
                0x3618, 0x00,
                0x3612, 0x29,
                0x3709, 0x52,
                0x370c, 0x03,
                0x3a02, 0x02,    // 60Hz max exposure
                0x3a03, 0xe0,    // 60Hz max exposure
                0x3a08, 0x00,    // B50 step
                0x3a09, 0x6f,    // B50 step
                0x3a0a, 0x00,    // B60 step
                0x3a0b, 0x5c,    // B60 step
                0x3a0e, 0x06,    // 50Hz max band
                0x3a0d, 0x08,    // 60Hz max band
                0x3a14, 0x02,    // 50Hz max exposure
                0x3a15, 0xe0,    // 50Hz max exposure
                0x4004, 0x02,    // BLC line number
```

```
                0x3002, 0x1c,   // reset JFIFO, SFIFO, JPG
                0x3006, 0xc3,   // disable clock of JPEG2x, JPEG
                0x4713, 0x03,   // JPEG mode 3
                0x4407, 0x04,   // Quantization sacle
                0x460b, 0x37,
                0x460c, 0x20,
                0x4837, 0x16,   // MIPI global timing
                0x3824, 0x04,   // PCLK manual divider
                0x5001, 0x83,   // SDE on, CMX on, AWB on
                0x3503, 0x00,   // AEC/AGC on
        };

        // Write preview table
        for (int i=0; i<sizeof(regVideo)/sizeof(int); i+=2)
        {
                OV5640_write_i2c(regVideo[i], regVideo[i+1]);
        }

        return 0;
}

int OV5640_capture_setting()
{
        // set OV5640 to capture mode
        int regCapture[] =
        {
                // YUV Capture
                // 2592 x 1944 3.75fps
                // 24 MHz input clock, 42Mhz PCLK
                0x3035, 0x71,   // PLL
                0x3036, 0x69,   // PLL
                0x3c07, 0x07,   // lightmeter 1 threshold[7:0]
                0x3820, 0x40,   // flip
                0x3821, 0x06,   // mirror
                0x3814, 0x11,   // timing X inc
                0x3815, 0x11,   // timing Y inc
                0x3800, 0x00,   // HS
                0x3801, 0x00,   // HS
                0x3802, 0x00,   // VS
                0x3803, 0x00,   // VS
                0x3804, 0x0a,   // HW (HE)
                0x3805, 0x3f,   // HW (HE)
                0x3806, 0x07,   // VH (VE)
                0x3807, 0x9f,   // VH (VE)
                0x3808, 0x0a,   // DVPHO
                0x3809, 0x20,   // DVPHO
                0x380a, 0x07,   // DVPVO
                0x380b, 0x98,   // DVPVO
                0x380c, 0x0b,   // HTS
                0x380d, 0x1c,   // HTS
                0x380e, 0x07,   // VTS
                0x380f, 0xb0,   // VTS
                0x3813, 0x04,   // timing V offset
                0x3618, 0x04,
                0x3612, 0x2b,
                0x3709, 0x12,
                0x370c, 0x00,
                0x4004, 0x06,   // BLC line number
                0x3002, 0x1c,   // reset JFIFO, SFIFO, JPG
                0x3006, 0xc3,   // disable clock of JPEG2x, JPEG
                0x4713, 0x02,   // JPEG mode 2
                0x4407, 0x0c,   // Quantization sacle
                0x460b, 0x37,
                0x460c, 0x20,
```

```
                0x4837, 0x2c,     // MIPI global timing
                0x3824, 0x01,     // PCLK manual divider
                0x5001, 0x83,     // SDE on, CMX on, AWB on, scale off
                0x3503, 0x03,      // AEC/AGC off
        };

        // Write capture table
        for (int i=0; i<sizeof(regCapture)/sizeof(int); i+=2)
        {
                OV5640_write_i2c(regCapture[i], regCapture[i+1]);
        }

        return 0;
}

int  OV5640_af_init()
{
        // download firmware
        // if supported, multiple bytes I2C writes are highly recommended.
        for (int i=0; i<sizeof(af_firmware)/sizeof(int); i+=2)
        {
                OVPantherDemo::WriteSCCB(0x78, af_firmware[i], af_firmware[i+1]);
        }

        return 0;
}

int  OV5640_auto_focus()
{
        int temp;
        // focus
                        OV5640_write_i2c(0x3022, 0x03);

        while(1)
        {
            // check status
            temp = OV5640_read_i2c(0x3029);
            if (temp ==0x10) return 0;   // focus completed
        }
         return 1;

        Delay(100);

}

int OV5640_get_sysclk()
{
        // calculate sysclk
        int temp1, temp2;
        int Multiplier, PreDiv, VCO, SysDiv, Pll_rdiv, Bit_div2x, sclk_rdiv, sysclk;

        int sclk_rdiv_map[] = {
                1, 2, 4, 8};

        temp1 = OV5640_read_i2c(0x3034);
        temp2 = temp1 & 0x0f;
        if (temp2 == 8 || temp2 == 10) {
                Bit_div2x = temp2 / 2;
        }

        temp1 = OV5640_read_i2c(0x3035);
        SysDiv = temp1>>4;
        if(SysDiv == 0) {
                SysDiv = 16;
```

```
        }

        temp1 = OV5640_read_i2c(0x3036);
        Multiplier = temp1;

        temp1 = OV5640_read_i2c(0x3037);
        PreDiv = temp1 & 0x0f;
        Pll_rdiv = ((temp1 >> 4) & 0x01) + 1;

        temp1 = OV5640_read_i2c(0x3108);
        temp2 = temp1 & 0x03;
        sclk_rdiv = sclk_rdiv_map[temp2];

        VCO = XVCLK * Multiplier / PreDiv;

        sysclk = VCO / SysDiv / Pll_rdiv * 2 / Bit_div2x / sclk_rdiv;

        return sysclk;
}

int OV5640_get_HTS()
{
        // read HTS from register settings
        int HTS;

        HTS = OV5640_read_i2c(0x380c);
        HTS = (HTS<<8) + OV5640_read_i2c(0x380d);

        return HTS;
}

int OV5640_get_VTS()
{
        // read VTS from register settings
        int VTS;

        VTS = OV5640_read_i2c(0x380e);
        VTS = (VTS<<8) + OV5640_read_i2c(0x380f);

        return VTS;
}

int OV5640_set_VTS(int VTS)
{
        // write VTS to registers
        int temp;

        temp = VTS & 0xff;
        OV5640_write_i2c(0x380f, temp);

        temp = VTS>>8;
        OV5640_write_i2c(0x380e, temp);

        return 0;
}

int OV5640_get_shutter()
{
        // read shutter, in number of line period
        int shutter;

        shutter = (OV5640_read_i2c(0x03500) & 0x0f);
        shutter = (shutter<<8) + OV5640_read_i2c(0x3501);
        shutter = (shutter<<4) + (OV5640_read_i2c(0x3502)>>4);
```

```
	return shutter;
}

int OV5640_set_shutter(int shutter)
{
	// write shutter, in number of line period
	int temp;

	shutter = shutter & 0xffff;

	temp = shutter & 0x0f;
	temp = temp<<4;
	OV5640_write_i2c(0x3502, temp);

	temp = shutter & 0xfff;
	temp = temp>>4;
	OV5640_write_i2c(0x3501, temp);

	temp = shutter>>12;
	OV5640_write_i2c(0x3500, temp);

	return 0;
}

int OV5640_get_gain16()
{
	// read gain, 16 = 1x
	int gain16;

	gain16 = OV5640_read_i2c(0x350a) & 0x03;
	gain16 = (gain16<<8) + OV5640_read_i2c(0x350b);

	return gain16;
}

int OV5640_set_gain16(int gain16)
{
	// write gain, 16 = 1x
	int temp;
	gain16 = gain16 & 0x3ff;

	temp = gain16 & 0xff;
	OV5640_write_i2c(0x350b, temp);

	temp = gain16>>8;
	OV5640_write_i2c(0x350a, temp);

	return 0;
}

int OV5640_get_light_frequency()
{
	// get banding filter value
	int temp, temp1, light_frequency;

	temp = OV5640_read_i2c(0x3c01);

	if (temp & 0x80) {
		// manual
		temp1 = OV5640_read_i2c(0x3c00);
		if (temp1 & 0x04) {
			// 50Hz
			light_frequency = 50;
```

```
            }
            else {
                // 60Hz
                light_frequency = 60;
            }
        }
        else {
            // auto
            temp1 = OV5640_read_i2c(0x3c0c);
            if (temp1 & 0x01) {
                // 50Hz
                light_frequency = 50;
            }
            else {
                // 60Hz
            }
        }
        return light_frequency;
}

void OV5640_set_bandingfilter()
{
        int preview_VTS;
        int band_step60, max_band60, band_step50, max_band50;

        // read preview PCLK
        preview_sysclk = OV5640_get_sysclk();

        // read preview HTS
        preview_HTS = OV5640_get_HTS();

        // read preview VTS
        preview_VTS = OV5640_get_VTS();

        // calculate banding filter
        // 60Hz
        band_step60 = preview_sysclk * 100/preview_HTS * 100/120;
        OV5640_write_i2c(0x3a0a, (band_step60 >> 8));
        OV5640_write_i2c(0x3a0b, (band_step60 & 0xff));

        max_band60 = int((preview_VTS-4)/band_step60);
        OV5640_write_i2c(0x3a0d, max_band60);

        // 50Hz
        band_step50 = preview_sysclk * 100/preview_HTS;
        OV5640_write_i2c(0x3a08, (band_step50 >> 8));
        OV5640_write_i2c(0x3a09, (band_step50 & 0xff));

        max_band50 = int((preview_VTS-4)/band_step50);
        OV5640_write_i2c(0x3a0e, max_band50);
}

int OV5640_set_AE_target(int target)
{
        // stable in high
        int fast_high, fast_low;
        AE_low = target * 23 / 25;   // 0.92
        AE_high = target * 27 / 25;  // 1.08

        fast_high = AE_high<<1;
        if(fast_high>255)
                fast_high = 255;

        fast_low = AE_low>>1;
```

```
        OV5640_write_i2c(0x3a0f, AE_high);
        OV5640_write_i2c(0x3a10, AE_low);
        OV5640_write_i2c(0x3a1b, AE_high);
        OV5640_write_i2c(0x3a1e, AE_low);
        OV5640_write_i2c(0x3a11, fast_high);
        OV5640_write_i2c(0x3a1f, fast_low);

        return 0;
}

int OV5640_init()
{
        // initialize OV5640
        OV5640_init_setting();

        return 0;
}

int OV5640_preview()
{
        // set OV5640 to preview mode
        OV5640_preview_setting();

        // calculate banding filter
        OV5640_set_bandingfilter();

        // set ae_target
        OV5640_set_AE_target(AE_Target);

        // update night mode setting
        OV5640_set_night_mode(m_iCombo_NightMode);

        // download auto focus firmware
        OV5640_af_init();

        return 0;
}

int OV5640_return_to_preview()
{
        // release focus
        OV5640_write_i2c(0x3022, 0x08);

        // set OV5640 to preview mode
        OV5640_preview_setting();

        // calculate banding filter
        OV5640_set_bandingfilter();

        // set ae_target
        OV5640_set_AE_target(AE_Target);

        // update night mode setting
        OV5640_set_night_mode(m_iCombo_NightMode);

        // re-launch auto focus zones
        OV5640_write_i2c(0x3022, 0x12);

        return 0;
}

int OV5640_video()
{
```

```
	// set OV5640 to video mode
	OV5640_video_setting();

	// calculate banding filter
	OV5640_set_bandingfilter();

	// set ae_target
	OV5640_set_AE_target(AE_Target);

	// turn off night mmode
	OV5640_set_night_mode(0);

	return 0;
}


int OV5640_capture()
{
	// set OV5640 to capture mode
	int preview_shutter, preview_gain16, average;
	int capture_shutter, capture_gain16;
	int capture_sysclk, capture_HTS, capture_VTS;
	int light_frequency, capture_bandingfilter, capture_max_band;
	long capture_gain16_shutter;

	//auto focus
	OV5640_auto_focus();

	// read preview shutter
	preview_shutter = OV5640_get_shutter();

	// read preview gain
	preview_gain16 = OV5640_get_gain16();

	// get average
	average = OV5640_read_i2c(0x56a1);

	// turn off night mode for capture
	OV5640_set_night_mode(0);

	// turn off overlay
	 OV5640_write_i2c(0x3022, 0x06);

	// Write capture setting
	OV5640_capture_setting();

	// read capture VTS
	capture_VTS = OV5640_get_VTS();
	capture_HTS = OV5640_get_HTS();
	capture_sysclk = OV5640_get_sysclk();

	// calculate capture banding filter
	light_frequency = OV5640_get_light_frequency();
	if (light_frequency == 60) {
		// 60Hz
		capture_bandingfilter = capture_sysclk * 100 / capture_HTS * 100 / 120;
	}
	else {
		// 50Hz
		capture_bandingfilter = capture_sysclk * 100 / capture_HTS;
	}
	capture_max_band = int((capture_VTS - 4)/capture_bandingfilter);

	// calculate capture shutter/gain16
```

```
	if (average > AE_low && average < AE_high) {
		// in stable range
		capture_gain16_shutter = preview_gain16 * preview_shutter * capture_sysclk/preview_sysclk *
preview_HTS/capture_HTS * AE_Target / average;
	}
	else {
		capture_gain16_shutter = preview_gain16 * preview_shutter * capture_sysclk/preview_sysclk *
preview_HTS/capture_HTS;
	}

	// gain to shutter
	if(capture_gain16_shutter < (capture_bandingfilter * 16)) {
		// shutter < 1/100
		capture_shutter = capture_gain16_shutter/16;
		if(capture_shutter <1)
			capture_shutter = 1;

		capture_gain16 = capture_gain16_shutter/capture_shutter;
		if(capture_gain16 < 16)
			capture_gain16 = 16;
	}
	else {
		if(capture_gain16_shutter > (capture_bandingfilter*capture_max_band*16)) {
			// exposure reach max
			capture_shutter = capture_bandingfilter*capture_max_band;
			capture_gain16 = capture_gain16_shutter / capture_shutter;
		}
		else {
			// 1/100 < capture_shutter =< max, capture_shutter = n/100
			capture_shutter = (int (capture_gain16_shutter/16/capture_bandingfilter)) * capture_bandingfilter;
			capture_gain16 = capture_gain16_shutter / capture_shutter;
		}
	}


	// write capture gain
	OV5640_set_gain16(capture_gain16);

	// write capture shutter
	if (capture_shutter > (capture_VTS - 4)) {
		capture_VTS = capture_shutter + 4;
		OV5640_set_VTS(capture_VTS);
	}
	OV5640_set_shutter(capture_shutter);

	// skip 2 vysnc

	// start capture at 3rd vsync

	return 0;
}
```

Revision History

Rev 2.01

Merge from “OV5640 Camera Module Software Application Notes” and “OV5640 Camera Module Hardware Application Notes”.

Rev 2.10

Add auto focus support.

Rev 2.11

Update contrast setting(as DB version A10/AM10).

Add 4.10 Scale.

Update demo driver. Move driver code from “Form1.h” to “PantherDemo.cpp”

R 2.12

Fixed capture exposure bug in driver code.

Add “write_i2c(0x4005, 0x1a); // BLC always update” in initial setting.

Update Initial Setting

R2.13

Add “4.3 Drive Capability” and “4.4 I/O control”