# OV2640摄像头模块

## 软件应用说明



上次修改时间：12月12日，2007文档

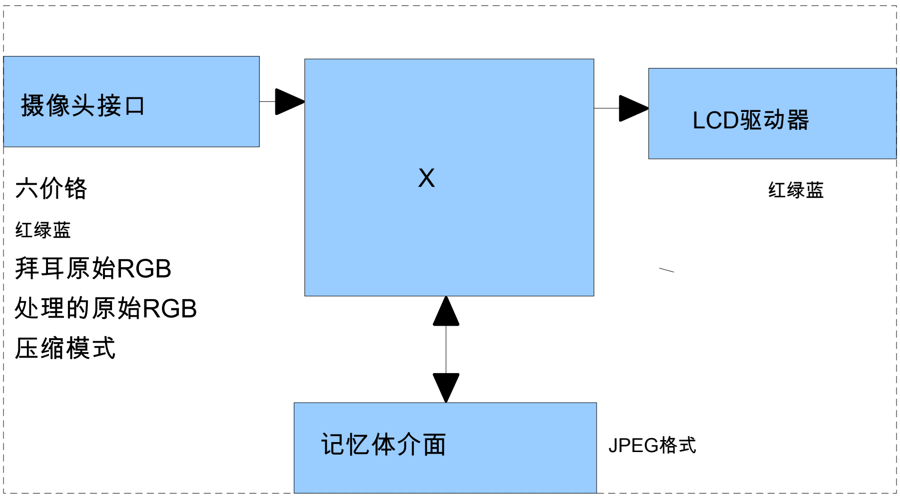
修订版：1.03

# 目录

## *1.如何选择输出格式？*

OV2640支持4种输出格式：YcbCr422，YCbCr420，RGB565，拜耳原始RGB和GRB，YUV422压缩。如何为照相手机设计或其他应用选择正确的输出格式？首先让我们看一下后端芯片。

后端芯片的总体示意图如下：




LCD驱动器上的数据格式始终为RGB。例如，RGB444，RGB565，RGB555，RGB888等。数据格式和存储接口始终为“压缩”。压缩数据是从YCbCr数据压缩而来的。因此，在后端芯片内部都需要RGB和YCbCr数据。对于不同的后端芯片，“ X”块是不同的。

### 1.1具有完整ISP的后端

这种后端具有完整的ISP。它需要原始RGB输入，进行插值以生成RGB24，并进行转换以生成YCbCr。这种后端可以采用拜耳原始RGB或经过处理的原始RGB。

与Bayer原始RGB相比，处理后的原始RGB的优势在于处理了输出数据。可以应用传感器功能，例如缺陷像素校正，镜头校正，伽玛，颜色矩阵，降噪，清晰度，BLC等。由于后端芯片的使用寿命比图像传感器更长，因此如果采用拜耳原始RGB，则有时后端芯片无法修复新传感器的缺陷。但是新传感器的缺陷可以在处理后的原始RGB输出中修复。

如果后端从传感器中获取拜耳原始RGB格式，则所有图像处理操作（例如缺陷像素校正，镜头校正，伽玛，颜色矩阵，降噪，清晰度，BCL等）都应由后端完成。如果后端从传感器获取已处理的原始RGB格式，则图像处理操作将包括缺陷像素校正，镜头校正，伽玛，颜色矩阵，去噪，

清晰度，BCL等可以在传感器内部或通过后端芯片完成。换句话说，用户可以选择在哪一侧进行图像处理操作。

## 1.2使用YCbCr ISP的后端

这种后端具有ISP，但只能采用YCbCr格式。ISP可以将YCbCr转换为RGB格式以用于LCD显示，并压缩YCbCr进行存储。

## 1.3没有ISP的后端

这种后端没有内置ISP。它不能通过硬件从一种格式转换为另一种格式。实际上，格式转换是通过软件完成的。这种后端芯片有3种可能的解决方案。

一种。传感器输出YCbCr。后端芯片将YCbCr转换为RGB，以通过软件显示。
b。传感器输出RGB565。后端芯片将RGB565转换为YCbCR进行压缩。
C。传感器输出RGB565用于预览，输出YCbCr用于捕获（压缩）。

解决方案 提供最佳的图像质量。由于输入数据是24位RGB等价的。可以将其转换为RGB888以用于LCD显示。解决方案b。提供最差的图像质量。由于输入数据仅为16位RGB565，因此即使将其转换为YCbCr，色深仍为16位。解决方案c。提供与解决方案类似的图像质量 但是由于预览为RGB565，捕获为YCbCr，因此预览图片可能看起来与捕获的图片略有不同。



## 1.4等式从一种格式转换为另一种格式

YCbCr转RGB24

Y = 0.299R + 0.587G + 0.114B

Cb = 0.568（BY）+ 128 = –0.172R – 0.339G + 0.511B + 128 Cr =

0.713（RY）+ 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; 铬=（（128

 * R-107 * G-21 * B）>> 8）+ 128;

RGB24至YcbCr

R = Y + 1.371（Cr – 128）

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

R = Y +（351 *（Cr – 128））>> 8

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

+（443 *（Cb – 128））>> 8

## *2.如何选择输出分辨率？*

### 2.1 ISP后端

如果后端芯片具有内置ISP（完整ISP或YCbCr ISP），则ISP可以进行图像缩放。因此，OV2640仅输出用于预览的SVGA格式和用于捕获的UXGA。ISP将SVGA图像缩放到移动设备用于LCD显示所需的其他分辨率。ISP将UXGA图像缩放到移动设备捕获所需的其他分辨率。

### 2.2没有ISP的后端

如果后端芯片不具备图像缩放功能，则必须使用OV2640的LCD缩放器来精确缩放LCD尺寸的输出分辨率。例如，如果LCD尺寸为176x220，则LCD缩放器会将输出尺寸缩放为176x220。

在这种情况下，OV2640输出较小的分辨率进行预览，并输出其他几种分辨率进行捕获。捕获的分辨率可能包括：QQVGA，QVGA，QCIF，CIF，VGA，SVGA，SXGA，UXGA。



## *3.如何调整帧频*

对于60Hz的光照环境，建议的帧速率为30fps和15fps预览；对于50Hz的光照环境，建议的帧率为25fps和14.3fps预览。对于60hz的光照环境，建议的捕获帧速率为7.5fps，对于50hz的光照环境，建议的帧速率为7.14fps。夜间模式的帧率较低，稍后我们将讨论夜间模式。

下面列出了上述帧速率的参考设置。

### 3.1 SVGA预览，30fps，24Mhz输入时钟

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

### 3.2 SVGA预览，15fps，24 Mhz输入时钟

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

### 3.3 SVGA预览，25fps，24 Mhz输入时钟

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

### 3.4 SVGA预览，14.3fps，24 Mhz输入时钟

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



### 3.5 UXGA Capture，7.5fps，24 Mhz输入时钟

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

### 3.6 UXGA Capture，7.14fps，24 Mhz输入时钟

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

## *4.如何设置夜间模式预览*

夜间模式有2种设置。例如，将一种类型设置为固定的低帧速率3.75fps。另一种类型设置为自动帧速率，例如从30fps到3.75fps。当环境明亮时，帧速率将增加到30fps。当环境黑暗时，帧速率降低到3.75fps。

### 4.1固定帧频的夜间模式

在60Hz光照环境下为3.75fps夜间模式，24Mhz时钟输入SCCB_salve_Address = 0x60;
write_SCCB（0xff，0x01）;
write_SCCB（0x11，0x07）;

在50Hz光照环境下为3.125fps夜间模式，24Mhz时钟输入SCCB_salve_Address = 0x60;
write_SCCB（0xff，0x01）;
write_SCCB（0x11，0x07）;

### 4.2具有自动帧频的夜间模式

30Hz～3.75fps夜间模式，适用于60Hz光照环境，24Mhz时钟输入SCCB_salve_Address = 0x60;
write_SCCB（0xff，0x01）;
write_SCCB（0x11，0x00）;
write_SCCB（0x0f，0x4b）;
write_SCCB（0x03，0xcf）;

15Hz～3.75fps夜间模式，适用于60Hz光照环境，24Mhz时钟输入SCCB_salve_Address = 0x60;
write_SCCB（0xff，0x01）;
write_SCCB（0x11，0x01）;
write_SCCB（0x0f，0x4b）;
write_SCCB（0x03，0x8f）;

在50Hz光照环境下为25fps～3.125fps夜间模式，24Mhz时钟输入SCCB_salve_Address = 0x60;

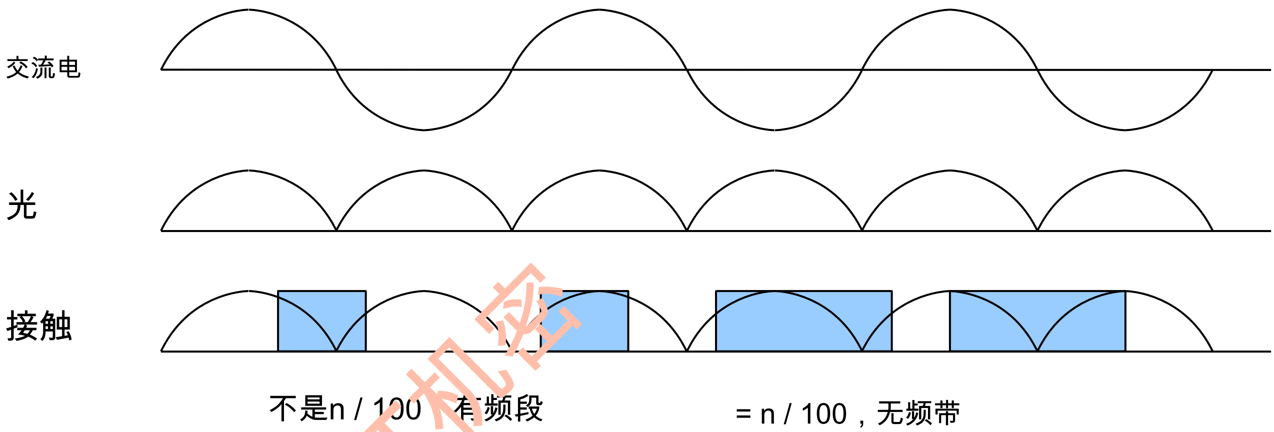
```
write_SCCB ( 0xff , 0x01 ) ;
write_SCCB ( 0x11 , 0x00 ) ;
write_SCCB ( 0x0f , 0x4b ) ;
write_SCCB ( 0x03 , 0xcf ) ;
```

在50Hz光照环境下为14.3fps~3.6fps夜间模式，24Mhz时钟输入SCCB_salve_Address = 0x60;

```
write_SCCB ( 0xff , 0x01 ) ;
write_SCCB ( 0x11 , 0x01 ) ;
write_SCCB ( 0x0f , 0x4b ) ;
write_SCCB ( 0x03 , 0x8f ) ;
```

## *5.如何在预览模式下删除光带*

### 5.1灯带




办公灯的强度不均匀。它随交流频率变化。例如，如果AC频率为50Hz，则光在100hz处改变强度。



## 5.2移除光带

通过将曝光设置为n / 100（n / 120表示60Hz）秒来去除光带。带状滤波器的值告诉OV2640多少行是1/100（1/120表示60Hz）秒。

## 5.3按地区信息选择条带过滤器

手机的区域信息可以用来选择带状滤波器的值。建立光频率表以指示哪个区域使用50Hz的光，哪个区域使用60 Hz的光。当获得区域信息时，可以从表中获得光频率信息。

不同的帧频可用于不同的光频率。因此，针对50hz光照条件和60hz光照条件均优化了帧速率。

用于30fps SVGA预览，24Mhz输入时钟SCCB_salve_Address = 0x60的带状滤
波器设置；
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; 位[5]启用带状滤波器
write_SCCB（0x0c，0x38）; //选择60hz频段过滤器，bit [1]控制自动，设置为0，关闭自动频段write_SCCB（0x4f，0xca）
; // 50Hz带状滤波器
write_SCCB（0x50，0xa8）; // 60Hz频段滤波器write_SCCB（0x5a，0x
23）; // 3步为50hz，4步为60hz

用于15fps SVGA预览，24Mhz输入时钟SCCB_salve_Address = 0x60的带状滤
波器设置；
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; 位[5]启用带状滤波器
write_SCCB（0x0c，0x38）; //选择60hz频段过滤器，位[1]控制自动，设置为0，关闭自动频段write_SCCB（0x4f，0x65）
; // 50Hz带状滤波器
write_SCCB（0x50，0x54）; // 60Hz频段滤波器write_SCCB（0x5a，0x
57）; // 6步为50hz，8步为60hz

用于25fps SVGA预览的带状滤波器设置，24Mhz输入时钟SCCB_salve_Addr
ess = 0x60;
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; 位[5]启用带状滤波器
write_SCCB（0x0c，0x3c）; //选择50hz频段过滤器，bit [1]控制自动，设置为0，关闭自动频段write_SCCB（0x4f，0xa8）
; // 50Hz带状滤波器
write_SCCB（0x50，0x8c）; // 60Hz频段滤波器write_SCCB（0x5a，0x
33）; // 4步为50hz，4步为60hz

用于14.3fps SVGA预览的频段过滤器设置，24Mhz输入时钟SCCB_salve_Addre
ss = 0x60;
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; 位[5]启用带状滤波器
write_SCCB（0x0c，0x3c）; //选择50hz频段过滤器，bit [1]控制自动，设置为0，关闭自动频段write_SCCB（0x4f，0x54）
; // 50Hz带状滤波器

write_SCCB（0x50，0x46）; // 60Hz频段滤波器write_SCCB（0x5a，0x
78）; // 8步为50hz，9步为60hz

## 5.4通过自动光频率检测选择带状滤波器

在50Hz和60Hz的光线环境下设置相同的帧频，并设置50Hz和60Hz的带状滤波器值。OV2640可以根据光频率检测自动选择50Hz或60Hz带状滤波器。

自动选择带状滤波器，用于30fps SVGA预览，24Mhz输入时钟SCCB_salve_Address = 0
x60；
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; [5]位使能带状滤波器write_SCCB（0x0c，0
x3a）; //自动选择条带过滤器write_SCCB（0x4f，0xca）; // 50Hz带状滤波
器
write_SCCB（0x50，0xa8）; // 60Hz频段滤波器write_SCCB（0x5a，0x
23）; // 3步为50hz，4步为60hz

用于15fsp SVGA预览，24Mhz输入时钟SCCB_salve_Address = 0x60的自动选择带状滤波
器；
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; [5]位使能带状滤波器write_SCCB（0x0c，0
x3a）; //自动选择条带过滤器write_SCCB（0x4f，0x65）; // 50Hz带状滤波
器
write_SCCB（0x50，0x54）; // 60Hz频段滤波器write_SCCB（0x5a，0x
57）; // 6步为50hz，8步为60hz

自动选择带状滤波器，用于25fps SVGA预览，24Mhz输入时钟SCCB_salve_Address = 0
x60；
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; [5]位使能带状滤波器write_SCCB（0x0c，0
x3a）; //自动选择条带过滤器write_SCCB（0x4f，0xa8）; // 50Hz带状滤波
器
write_SCCB（0x50，0x8c）; // 60Hz频段滤波器write_SCCB（0x5a，0x
33）; // 4步为50hz，4步为60hz



用于14.3fsp SVGA预览，24Mhz输入时钟SCCB_salve_Address = 0x60的自动选择带状滤波
器；
write_SCCB（0xff，0x01）; //选择bank1
write_SCCB（0x13，0xe5）; [5]位使能带状滤波器write_SCCB（0x0c，0
x3a）; //自动选择条带过滤器write_SCCB（0x4f，0x60）; // 50Hz带状滤波
器
write_SCCB（0x50，0x50）; // 60Hz频段滤波器write_SCCB（0x5a，0x
67）; // 7步为50hz，8步为60hz

## 5.5删除捕获中的光带

如果捕获使用与预览相同的分辨率和帧速率，则在

捕获。如果捕获使用不同的分辨率或不同的帧频作为预览，则应通过曝光计算在捕获中删除光带。请阅读14.7.3节。

## 5.6无法去除灯带时

通常情况下，光带会通过带状滤光片去除。

但是有一些特殊条件，例如太阳光和办公室灯的混合光，拍摄荧光灯，不能去除光带。原因是对于50hz的光照环境，曝光时间小于1/100秒，对于60hz的光照环境，曝光时间小于1/120秒，因此无法去除光带。

不能在所有CMOS传感器上消除此条件的光带，不仅是OV2640。因此，在这种情况下无法消除光带。

# *6.白平衡*

OV2640支持简单的白平衡和高级平衡。

## 6.1简单白平衡

简单的白平衡假设为“灰色世界”。这意味着世界的平均颜色是灰色。对于大多数环境都是如此。

简单的运单的优势

简单的白平衡不取决于镜头。简单白平衡的一般设置可以应用于具有不同镜头的所有模块。

简单AWB的缺点

在“灰色世界”不正确的情况下，颜色不正确。例如，背景具有巨大的红色，蓝色或绿色等。前景的颜色不准确。如果相机将单色，红色，蓝色，绿色作为目标，那么简单的白平衡将使单色变为灰色。



设定值

SCCB_salve_Address = 0x60;
write_SCCB（0xff，0x00）; //选择bank0 write_SCCB
（0xc7，0x10）; //简单的运单

## 6.2高级白平衡

高级白平衡使用色温信息检测白色区域并进行白平衡。

先进的AWB的优势

颜色比简单的白平衡更准确。即使背景是单色，相机也不会将单色变成灰色。

先进的AWB的缺点

高级白平衡设置取决于镜头。必须为每个装有新镜头的模块调整设置。必须由光学实验室的OmniVision FAE使用某些光学设备（例如灯箱，颜色检查器等）进行调整。

设定值

与OmniVision本地FAE联系。

## 6.3如何选择？

通常，对于低分辨率相机模块（如CIF，VGA和1.3M），选择简单的AWB。对于2M，3M等高分辨率相机模块，选择了高级AWB。

## *7.缺陷像素校正*

缺陷像素包括死像素和受伤像素。

坏点包括白色坏点和黑色坏点。无论实际图像是亮还是暗，白色的死像素始终是白色的。无论实际图像是亮还是暗，黑死像素始终为黑色。

受伤的像素可能会随光线而变化，但不如正常像素大。受伤的白色像素比正常像素明亮得多，但不是完整的白色。黑色受伤的像素比正常像素要暗得多，但不是完整的黑色。

OV2640具有内置的缺陷像素校正功能。如果OV2640输出YCbCr，RGB565，已处理的原始RGB，则可以启用缺陷像素校正功能以修复缺陷像素。但是，如果使用Bayer raw RGB，则无法使用传感器的缺陷像素校正功能。应该使用后端芯片的缺陷像素校正。



请注意后端芯片的缺陷像素校正功能。某些后端芯片可能无法校正OV2640的所有缺陷像素。

## *8. BLC*

黑电平校准（BLC）的功能是在图像的暗区产生准确的颜色。OV2640内置有自动BLC功能。应该始终将其打开。

## *9.视频模式*

视频模式需要高帧速率，通常固定为15fps。视频模式没有夜间模式。

## *10.数码变焦*

如果OV2640输出的图像小于SVGA，则可能支持连续数字变焦。例如

| | |
|---|---|
| UXGA | 不支持数字变焦 |
| SVGA | 1-2倍 |
| 显卡 | 1-2.5倍 |
| QVGA | 1-5倍 |

如果后端芯片支持扩大，则可以支持更多的缩放级别。

## *11. OV2640功能*

### 11.1灯光模式

汽车

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

阳光明媚

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

多云的

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



办公室

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

家

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

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

## 11.2颜色饱和度

OV2640的色彩饱和度可以调节。高的色彩饱和度会使图片看起来更生动，但是副作用是更大的噪点，而且肤色不准确。

饱和度+ 2
SCCB_salve_Address = 0x60;
write_SCCB（0xff，0x00）;
write_SCCB（0x7c，0x00）;
write_SCCB（0x7d，0x02）;
write_SCCB（0x7c，0x03）;
write_SCCB（0x7d，0x68）;
write_SCCB（0x7d，0x68）;

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



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

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

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

## 11.3亮度

OV2640的亮度可以调节。较高的亮度将使图像更亮。较高亮度的副作用是图像看起来模糊。

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

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

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

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

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

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

## 11.4对比

OV2640的对比度可以调整。较高的对比度将使图像清晰。但是副作用是失去了动态范围。

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

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

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

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

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

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

## 11.5特效

OV2640支持某些特殊效果，例如黑白，负片，棕褐色，带蓝色，带红色，带绿色等。如果用户需要其他特殊效果，则应使用后端芯片支持。



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

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

浅绿色

SCCB_salve_Address = 0x60;

write_SCCB ( 0xff , 0x00 ) ;

write_SCCB ( 0x7c 0x00 ) ;

write_SCCB ( 0x7d , 0x18 ) ;

write_SCCB ( 0x7c 0x05 ) ;

write_SCCB ( 0x7d , 0x40 ) ;

write_SCCB ( 0x7d , 0x40 ) ;

带红色

SCCB_salve_Address = 0x60;

write_SCCB ( 0xff , 0x00 ) ;

write_SCCB ( 0x7c 0x00 ) ;

write_SCCB ( 0x7d , 0x18 ) ;

write_SCCB ( 0x7c 0x05 ) ;

write_SCCB ( 0x7d , 0x40 ) ;

write_SCCB ( 0x7d , 0xc0 ) ;

黑白

SCCB_salve_Address = 0x60;

write_SCCB ( 0xff , 0x00 ) ;

write_SCCB ( 0x7c 0x00 ) ;

write_SCCB ( 0x7d , 0x18 ) ;

write_SCCB ( 0x7c 0x05 ) ;

write_SCCB ( 0x7d , 0x80 ) ;

write_SCCB ( 0x7d , 0x80 ) ;



消极的

SCCB_salve_Address = 0x60;

write_SCCB ( 0xff , 0x00 ) ;

write_SCCB ( 0x7c 0x00 ) ;

write_SCCB ( 0x7d , 0x40 ) ;

write_SCCB ( 0x7c 0x05 ) ;

write_SCCB ( 0x7d , 0x80 ) ;

write_SCCB ( 0x7d , 0x80;

黑白负面

SCCB_salve_Address = 0x60;

write_SCCB ( 0xff , 0x00 ) ;

write_SCCB ( 0x7c 0x00 ) ;

write_SCCB ( 0x7d , 0x58 ) ;

write_SCCB ( 0x7c 0x05 ) ;

write_SCCB ( 0x7d , 0x80 ) ;

write_SCCB ( 0x7d , 0x80;

普通的

SCCB_salve_Address = 0x60;

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

## 11.6 YUV序列

**YUYV**

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

temp = read_SCCB ( 0xda ) ;            //将0xda的位0注册为'0'
温度 & = 0xfe;
write_SCCB ( 0xda , temp ) ;

temp = read_SCCB ( 0xda ) ;            //将0xc2的位4注册为'0'
温度 & = 0xef;
write_SCCB ( 0xda , temp ) ;
```

**V**

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

temp = read_SCCB ( 0xda ) ;            //将0xda的位0注册为'0'
温度 & = 0xfe;
write_SCCB ( 0xda , temp ) ;

temp = read_SCCB ( 0xda ) ;            //将0xc2的位4注册为'1'
温度| = 0x10;
write_SCCB ( 0xda , temp ) ;
```

**维尤**

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

temp = read_SCCB ( 0xda ) ;            //将0xda的位0注册为'1'
温度| = 0x01;
write_SCCB ( 0xda , temp ) ;

temp = read_SCCB ( 0xda ) ;            //将0xc2的位4注册为'0'
温度 & = 0xef;
write_SCCB ( 0xda , temp ) ;
```

**尤维**
SCCB_slave_address = 0x60;
write_SCCB（0xff，0x00）;

temp = read_SCCB（0xda）;　　　　//将0xda的位0注册为'1'
温度L = 0x01;
write_SCCB（0xda，temp）;

temp = read_SCCB（0xda）;　　　　//将0xc2的位4注册为'0'
温度| = 0x10;
write_SCCB（0xda，temp）;

## *12.与镜头打交道*

### 12.1光线掉落

光线掉落意味着图像的一角比图像的中心暗。这是由镜头引起的。可以打开OV2640的镜头阴影校正功能以补偿角落亮度，并使整个图像看起来一样明亮。

### 12.2暗角

某些镜头可能有暗角。暗角表示图片的颜色几乎为黑色。无法通过镜头校正来校正暗角。因此，带有暗角的模块为NG，无法使用。

### 12.3分辨率



摄像头模块的分辨率取决于镜头设计，聚焦调整和传感器分辨率。聚焦调整对于相机模块的组装非常重要。

对于OV2640，聚焦距离约为120〜150cm。景深约为60〜75厘米至无限。如果要检查摄像头模块的分辨率，则分辨率表应放置在120〜150 cm处。

### 12.4光学对比

镜头的光学对比度对图像质量非常重要。如果镜头的光学对比度不好，则图像会显得模糊。尽管可以通过增加传感器对比度使图像更锐利来改善它，但是较高的传感器对比度将使图像暗区的细节丢失。

### 12.5镜头盖

镜头盖是光路中最便宜的部分。但这可能会极大地影响图像质量。镜头盖应由两侧都镀有增透膜的光学玻璃制成。否则，镜头

眩光增强。

## 13.参考设置

## 13.1 YCbCr参考设置

### 13.1.1 VGA预览

// OV2640_SVGA_YUV_AM 14.3 fps // 24 MH
z输入时钟
//
//
SCCB_slave_address = 0x60;
write_SCCB ( 0xff , 0x01 ) ;
write_SCCB ( 0x12 , 0x80 ) ;
延迟 ( 1ms ) ;
write_SCCB ( 0xff , 0x00 ) ;
write_SCCB ( 0x2c , 0xff ) ;
write_SCCB ( 0x2e , 0xdf ) ;
write_SCCB ( 0xff , 0x01 ) ;
write_SCCB ( 0x3c , 0x32 ) ;
//
write_SCCB ( 0x11 , 0x01 ) ;
write_SCCB ( 0x09 , 0x02 ) ;
write_SCCB ( 0x04 , 0x28 ) ;
write_SCCB ( 0x13 , 0xe0 ) ;
write_SCCB ( 0x14 , 0x48 ) ;
write_SCCB ( 0x2c , 0x0c ) ;
write_SCCB ( 0x33 , 0x78 ) ;
write_SCCB ( 0x3a , 0x33 ) ;
write_SCCB ( 0x3b , 0xfB ) ;
//
write_SCCB ( 0x3e , 0x00 ) ;
write_SCCB ( 0x43 , 0x11 ) ;
write_SCCB ( 0x16 , 0x10 ) ;
//
write_SCCB ( 0x39 , 0x92 ) ;
//
write_SCCB ( 0x35 , 0xda ) ;
write_SCCB ( 0x22 , 0x1a ) ;
write_SCCB ( 0x37 , 0xc3 ) ;
write_SCCB ( 0x23 , 0x00 ) ;
write_SCCB ( 0x34 , 0xc0 ) ;
write_SCCB ( 0x36 , 0x1a ) ;
write_SCCB ( 0x06 , 0x88 ) ;

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

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

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

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

#### *13.1.2 UXGA捕获*

```
// OV2640_UXGA_YUV_AM 7.5 fps // 24 MH
z输入时钟
//
SCCB_slave_address = 0x60;
write_SCCB ( 0xff , 0x01 ) ;

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

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

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

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

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

## 13.2 RGB 565参考设置

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



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

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



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

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

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



### 13.3 RGB原始参考设置

联系OmniVision本地FAE

## *14.捕获序列*

## 14.1快门

OV2640的快门控制曝光时间。快门的单位是行周期。

快门值对于每个输出分辨率都有限制。如果未插入虚拟线，则QXGA分辨率的最大快门值为1248。XGA分辨率的最大快门值为672。

Default_XGA_maximum_shutter = 672;
Default_QXGA_maximum_shutter = 1248;

快门值存储在存储体1的3个寄存器reg0x45，reg0x10和reg0x04中。快门=（reg0x45＆0x3f）<<

10 + reg0x10 << 2 +（reg0x04＆0x03）;

## 14. 2条假人线

可以通过插入虚拟线来进一步增加曝光量。当插入虚拟线时，帧频也会改变。

可以插入2种虚拟线。数据输出前的虚拟线和数据输出后的虚拟线。

### *14.2.1额外的行*

如果在数据输出之前插入虚拟线（称为多余线），则实际曝光时间会增加。额外的行由存储体1的寄存器0x2d和0x2e控制。

曝光=快门+ Extra_lines Extra_lines = reg0x2d +

（reg0x2e << 8）;



最大快门值不变。

因此，即使快门值为0，最小曝光时间也为Extra_lines。
通常，应自动插入多余的行。如果环境较暗，则需要更长的曝光时间，因此会增加额外的行数。如果环境明亮，则需要较短的曝光时间，减少了额外的行数。如果手动插入多余的行且其值是固定的，则总的曝光时间不能少于明亮的环境中的Extra_line，否则图像将曝光过度。

多余的线插入到Vsync的有效期内，不更改Vsync无效的输出期的时间。

### *14.2.2虚拟线*

如果在数据输出后插入虚拟线（称为虚拟线），则最大快门

值已更改。虚设线插入两个Vsync之间。哑线的数量由存储体1的寄存器0x46和0x47控制。

SVGA_maximum_shutter =默认_SVGA_Maximum_Shutter + Dummy_line UXGA_maximum_shutter = Default_UXGA_Maximum_Shutter + Dummy_line

曝光时间为曝光=快门

Dummy_line = Reg0x47 << 8 + Reg0x46;

| | 多余的线 | 虚拟线 |
|---|---|---|
| 登记册 | 0x2d，0x2e | 0x46、0x47 |
| 最小快门值 | 0 | 0 |
| 最大快门值 | UXGA为1248<br>适用于SVGA的672 | 1248 +用于UXGA 672的Dummy_line +用于SVGA的Dummy_line |
| 最低暴露量 | Extra_lines | 0 |
| 最大曝光 | Maximum_Shutter + Extra_lines Maximum_Shutter | |

因此，如果同时插入了虚拟线和多余线，则曝光时间为曝光=快门+多余线

并且最大快门值为

SVGA_maximum_shutter =默认_SVGA_Maximum_Shutter + Dummy_line UXGA_maximum_shutter = Default_UXGA_Maximum_Shutter + Dummy_line



## 14. 3个虚拟像素

如果未插入虚拟像素，则线宽称为默认线宽。Default_SVGA_Line_Width = 1190;

Default_UXGA_Line_Width = 1922;

插入虚拟像素时，线宽会发生变化，帧频也会发生变化。

SVGA_Line_Width =默认值_SVGA_Line_Width + Dummy_pixel UXGA_Line_Width =默认值_UXGA_Line_Width + Dummy_pixel

## 14.4增益

增益存储在存储体1的reg0x00和Reg0x45 [7：6]中。如果仅使用Reg0x00的增益，则可以达到32x的最大增益。照相手机就足够了。因此，我们在这里不讨论reg0x45。

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

## 14.5带状滤波器

### *14.5.1预览*

自动条带过滤器用于预览。

### *14.5.2捕获*

手动条带过滤器用于捕获。带状滤波器的计算公式为

对于50Hz，带状滤波器计算为

Banding_Filter = Capture_PCLK_Frequency / 100 / capture_line_width

对于60Hz，带状滤波器计算为

Banding_Filter = Capture_PCLK_Frequency / 120 / capture_line_width So

Capture_Exposure = n * Banding_Filter

n是整数。

## 14.6自动帧频

可以通过打开夜间模式来启用自动帧频。启用夜间模式后，多余的线会自动调整。



## 14.7捕获顺序

### *14.7.1预览*

//初始化OV2640进行预览

//可以插入虚拟像素和虚拟行以进行预览Preview_dummy_pixel =

Preview_dummy_line =

Reg0x2b = Preview_dummy_pixel_reg＆0x00ff; Reg0x2a =
SCCB_read（0x2a）;
Reg0x2a = Reg0x2a和0x0f |（Preview_dummy_pixel_reg＆0x0f00）>> 4 SCCB_write（0x2a，R
eg0x2a）;
SCCB_write（0x2b，Reg0x2b）;

//更新哑行
Reg0x46 = Preview_dummy_line & 0x00ff; Reg0x47 =
Preview_dummy_line >> 8;
SCCB_write ( 0x46 , Reg0x46 ) ;
SCCB_write ( 0x47 , Reg0x47 ) ;

***14.7.2停止预览***

//停止AE / AG

reg0x13 = SCCB_read ( 0x13 ) ;
Reg0x13 = Reg0x13 & 0xfa;
SCCB_write ( 0x13 , reg0x13 ) ;

//读回预览快门reg0x45 = SCCB_re
ad ( 0x45 ) ;
reg0x10 = SCCB_read ( 0x10 ) ;
reg0x04 = SCCB_read ( 0x04 ) ;

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

//读回多余的行
reg0x2d = SCCB_read ( 0x2d ) ;
reg0x2e = SCCB_read ( 0x2e ) ;
Extra_lines = reg0x2d + ( reg0x2e << 8 ) ;

Preview_Exposure =快门+多余的线条；

//读取预览预览的回增益reg0x00 = S
CCB_read ( 0x00 ) ;
Preview_Gain16 = ( ( ( ( Reg0x00 & 0x[illegible]0 ) >> 4 ) + 1 ) * ( 16 + reg0x00 & 0x0f ) ;

//读回虚拟像素reg0x2a = SCCB_re
ad ( 0x2a ) ;
reg0x2b = SCCB_read ( 0x2b ) ;
Preview_dummy_pixels = ( reg0x2a & 0xf0 ) << 4 + reg0x2b;



***14.7.3计算捕获暴露***

//可以插入虚拟像素和虚拟行进行捕获Capture_dummy_pixel =

Capture_dummy_line =
Preview_PCLK_frequency =
Capture_PCLK_frequency =

//可以定义捕获最大增益。
// Capture_max_gain16 = capture_max_gain * 16
Capture_max_gain16 =
Preview_line_width = Default_SVGA_Line_Width + Preview_dummy_pixel;

```
如果（分辨率== SVGA）{
        Capture_line_width = Default_SVGA_Line_Width + capture_Dummy_pixel; }

别的 {
        Capture_line_width = Default_UXGA_Line_Width + capture_Dummy_pixel;
}

如果（分辨率== SVGA）{
        Capture_maximum_shutter =默认值_SVGA_maximum_shutter + capture_dummy_lines; }

别的 {
        Capture_maximum_shutter =默认值_UXGA_maximum_shutter + capture_dummy_line;
}

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

//计算带状滤波器值If（50Hz）{

        如果（格式== RGB）{// RGB表示原始RGB
                Capture_banding_Filter = Capture_PCLK_Frequency / 100 / capture_line_width; }

        别的 {
                Capture_banding_Filter = Capture_PCLK_Frequency / 100 /（2 * capture_line_width）;
        }
别的 {
        如果（格式== RGB）{
                Capture_banding_Filter = Capture_PCLK_Frequency / 120 / capture_line_width; }

        别的 {
                Capture_banding_Filter = Capture_PCLK_frequency / 120 /（2 * capture_line_width）;
        }
}

//重新分配收益和风险
Gain_Exposure = Preview_Gain16 * Capture_Exposure; 如果（Gain_Exp
osure <Capture_Banding_Filter * 16）{
        //曝光<1/100
        Capture_Exposure =增益曝光/ 16;
        Capture_Gain16 =（Gain_Exposure * 2 +1）/ Capture_Exposure / 2;
}
别的 {
        如果（Gain_Exposure> Capture_Maximum_Shutter * 16）{
                //曝光> Capture_Maximum_Shutter Capture_Exposure = C
                apture_Maximum_Shutter;
                Capture_Gain16 =（Gain_Exposure * 2 +1）/ Capture_Maximum_Shutter / 2;
```

```
                如果（Capture_Gain16> Capture_Max_Gain16）{
                        //达到最大，插入额外的行
                        Capture_Exposure = Gain_Exposure * 1.1 / Capture_Max_Gain16; //曝光= n / 1
                        00
                        Capture_Exposure = Gain_Exposure / 16 / Capture_banding_filter;
                        Capture_Exposure =
                                Capture_Exposure * Capture_banding_filter;
                        Capture_Gain16 =（Gain_Exposure * 2 + 1）/ Capture_Exposure / 2;
                }
        }
        别的 {
                // 1/100 <曝光<Capture_Maximum_Shutter，曝光= n / 100 Capture_Exposure =增益
                曝光/ 16 / Capture_banding_filter; Capture_Exposure = Capture_Exposure * Capture
                _banding_filter; Capture_Gain16 =（Gain_Exposure * 2 +1）/ Capture_Exposure /
                2;
        }
}
```

***14.7.4切换到UXGA***

//写入寄存器，更改为UXGA分辨率。

***14.7.5写寄存器***

//写哑像素

```
Reg0x2b = Capture_dummy_pixel_reg & 0x00ff; Reg0x2a =
SCCB_read（0x2a）;
Reg0x2a =（Reg0x2a和0x0f）|（（Capture_dummy_pixel_reg & 0x0f00）>> 4）; SCCB_write（0x2a，R
eg0x2a）;
SCCB_write（0x2b，Reg0x2b）;
```

//写虚拟行

```
Reg0x46 = Capture_Dummy_lines & 0x00ff; Reg0x47 =
Capture_Dummy_lines >> 8;
SCCB_write（0x46，Reg0x46）;
SCCB_write（0x47，Reg0x47）;
```

//写曝光

```
如果（Capture_Exposure> Capture_maximum_shutter）{
        快门= Capture_maximum_shutter;
        Extra_lines = Capture_Exposure – Capture_maximum_shutter;
}
别的 {
        快门= Capture_Exposure;
        Extra_lines = 0;
}
Reg0x04 = SCCB_read（0x04）;
Reg0x04 = Reg0x04和0xfc |（快门 & 0x000003）;
```

```
Reg0x10 = ( 快门>> 2 ) & 0x00ff; Reg0x45 = S
CCB_read ( 0x45 ) ;
Reg0x45 = ( reg0x45 & 0xc0 ) | ( ( 快门>> 10 ) & 0x3f ) ;

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

//写多余的行
Reg0x2d = Extra_lines & 0x00ff; Reg0x2e =
Extra_lines >> 8;
SCCB_write ( 0x2d , Reg0x2d ) ;
SCCB_write ( 0x2e , Reg0x2e ) ;

//写增益
增益= 0;
如果 ( Capture_Gain16> 16 ) {
        Capture_Gain16 = Capture_Gain / 2; 增益= 0x
        10;
}
如果 ( Capture_Gain16> 16 ) {
        Capture_Gain16 = Capture_Gain / 2; 增益=增
        益| 0x20;
}
如果 ( Capture_Gain16> 16 ) {
        Capture_Gain16 = Capture_Gain / 2; 增益=增
        益| 0x40;
}
如果 ( Capture_Gain16> 16 ) {
        Capture_Gain16 = Capture_Gain / 2; 增益=增
        益| 0x80;
}

增益=增益| ( Capture_Gain16 -16 ) ; SCCB_write

( 0x00 , 增益 ) ;
```



### ***14.7.6捕获***

//等待2个Vsync //捕获3个 rd 框
架。

### ***14.7.7返回预览***

//写入寄存器，更改为SVGA

。。。

//启动AG / AE

```
Reg0x13 = SCCB_read ( 0x13 ) ;

Reg0x13 = Reg0x13 | 0x05;

SCCB_Write ( 0xff , 0x01 ) ;

SCCB_Write ( 0x13 , Reg0x13 ) ;
```