# OV7675 照相模组

# 软件应用指南



修改日期：2009-07-30

版本：1.02

# 1. OmniVision 认证模组/镜头列表(采用 OV7675)

## *1.1 认证模组列表（2009 年 7 月 30 日）*

| OV7675版本 | 模组尺寸 | 镜头型号 | 模组厂 | 模组型号 | 接口 |
|---|---|---|---|---|---|
| R1A | 6.5*6.5*3.8 | S3028 | Sunny | 8079J | 标准PIN |
| R1A | 6.0*6.0*3.8 | S3028 | Sunny | 8079G | 标准PIN |
| R1A | 6.0*6.0*3.8 | S3028 | A-Kerr | 7675FSL R1.0(090423) | 标准PIN |
| R1A | 6.0*6.0*3.8 | CA513 | Darling | DL030-OV7675A | 标准PIN |
| R1A | 6.0*6.0*3.8 | CA513 | Truly | 8190 | 标准PIN |
| R1A | 6.0*6.0*3.8 | CA513 | Sunrise | PCV767501A | 标准PIN |
| R1A | 6.0*6.0*3.8 | CA513 | Foxconn | I059 | 标准PIN |

## *1.2 推荐镜头列表（2009 年 7 月 30 日）*

| 镜头厂 | 镜头型号 | 镜头结构 | 螺牙 | 相对孔径 | 光学总长 | 视场角 | 最大像面 | 模组尺寸 |
|---|---|---|---|---|---|---|---|---|
| Hokuang | CA513 | 2P+IR CUT | M5*0.35P | 2.8 | 3.1 | 60 | 2.3 | 6.0*6.0*3.8 |
| Sunny | S3028 | 2P+IR CUT | M5*0.35P | 2.8 | 3.1 | 62 | 2.2 | 6.0*6.0*3.8 |
| Largan | 9240A | 2P+IR CUT | M3.5*0.35P | 2.8 | 2.04 | 65 | 2.3 | 4.5*4.5*2.8 |
| BASO | LM002 | 2P+IR CUT | M4*0.25P | 2.8 | 3 | 59 | 2.32 | 6.0*6.0*3.7 |

## 2. OV7670 兼容模式

OV7675 支持两种工作模式：OV7670 兼容模式和 OV7675 模式。OV7675 的缺省模式为 OV7670 兼容模式。

在 OV7670 兼容模式下，OV7675 可以在大部分为 OV7670 写的驱动程序下工作。但是 OV7670 兼容模式并不能保证在所有为 OV7670 写的驱动程序下都能正常工作。如果 OV7675 不能在用户的 OV7670 驱动程序下工作，请使用 OV7675 模式并联系 OmniVision 当地 FAE。在 OV7670 兼容模式下， 有一些寄存器的参数是预先调好的，用户不能修改。OV7675 模组必须采用 OmniVision 推荐的镜头，并且经过 OmniVision 认证，才能保证在 OV7670 兼容模式下正常工作。OmniVision 认证的模组列表请参见 1.1。

OmniVision 每月会更新 OV7675 认证模组列表和推荐镜头列表。本文挡所附列表可能不是最新的。请与 OmniVision 当地 PM 确认最新认证模组列表和推荐镜头列表。

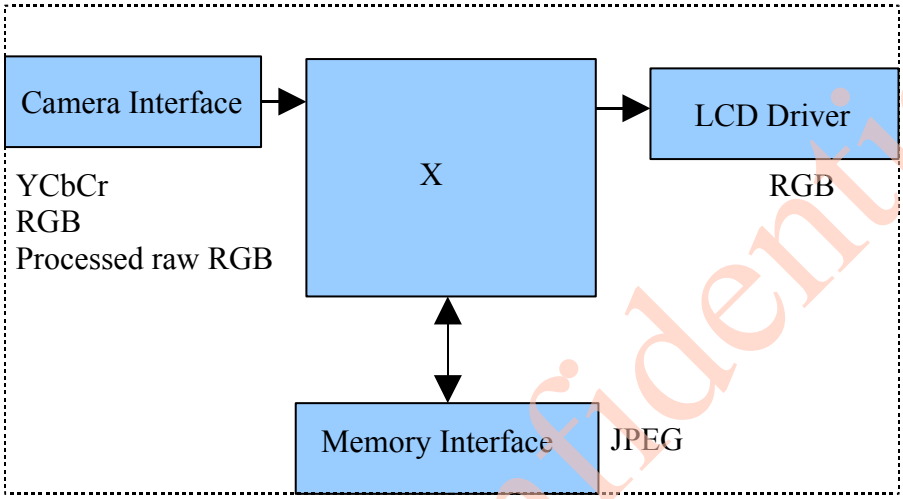
本文挡中所有图片都是在 OV7670 兼容模式下拍摄的。

在 OV7675 模式下，所有的寄存器参数都可以调整。如果 OV7675 不能在用户的 OV7670 驱动程序下很好运行，或者在用户的项目中不打算兼容 OV7670, 或者用户希望得到比 OV7675 兼容模式更好的图像品质（当然需要用户使用的模组可以提供更好的图像品质），用户可以采用 OV7675 模式，请联系 OmniVision 当地 FAE 帮助调试。

# 3. 选择输出格式

OV7675 支持的输出格式: YCbCr, RGB565, CCIR656 and RAW RGB. 在手机设计或其它应用中如何选择输出格式，首先看看以下 backend （后端）芯片框图：

常见后端芯片框图如下：




LCD 接收的数据格式是 RGB. 如, RGB444, RGB565, RGB555, RGB888 等数据格式，存储器接口总是压缩格式. 压缩的数据是从 YCbCr 数据压缩而来. 因此 RGB 和 YCbCr 数据都要放在芯片中. 不同的后端芯片“X”块是不同的.

## *3.1 有完整 ISP 的后端*

这是有完整 ISP 的后端.它可接收 raw RGB 数据, 做插补得到 RGB24 数据，再做转换得到 YCbCr 数据. 这种后端芯片可接收 Bayer raw RGB 或 processed raw RGB 数据.

processed raw RGB 是在 Bayer raw RGB 基础上经过处理的输出数据. Sensor 的一些功能如：死伤点校正、镜头补偿、gamma 校正、去噪点、锐度、暗电流补偿等都可做。如果用 Bayer raw RGB，有时后端芯片不能处理图像传感器的一些缺陷，但如果用 processed raw RGB 数据，许多缺陷已在图像传感器中处理好。

如果后端接收的是 Bayer raw RGB，那所有的图像处理由它来做，如死伤点校正、镜头补偿、gamma 校正、去噪点、锐度等，除了暗电流补偿。 如果后端接收的是 processed raw RGB 数据，那图像处理，如死伤点校正、镜头补偿、gamma 校正、去噪点、锐度等可在图像传感器里做，也可在后端芯片做。换句话说，就是用户可选择图像处理由那边来做。

## 3.2 后端有 YCbCr ISP

这种类型的后端芯片有 ISP，但只能接 YCbCr 数据，ISP 可把 YCbCr 数据转成 RGB 给 LCD 显示，也可转成压缩过的 YCbCr 以供存储。

## 3.3 后端没有 ISP

这种类型的后端芯片没有 ISP。它不能通过硬件把数据从一种格式转成另一种格式，实际数据格式的转换是通过软件来完成。对这种后端芯片有三种解决方案：

a. Sensor 输出 YCbCr. 后端通过软件把 YCbCr 转成 RGB 用于显示。
b. Sensor 输出 RGB565. 后端把 RGB565 转成 YCbCr 用于压缩处理。
c. Sensor 输出 RGB565 预览, 输出 YCbCr 拍照 ( 压缩)。

方案 a.最好的图像质量，输入的数据是 24-bit RGB，它转换成 RGB888 在 LCD 上显示
b.较差的图像质量，输入的数据是 16-bit RGB565，又转换成 YCbCr，色域仍是 16-bit。
c. 拍照的图像质量和方案 a 相同，但预览还是 RGB565，拍照是 YCbCr，预览图像和拍照图像看起来会有些不一样。

## 3.4 从一种输出格式到另一种输出格式的转换关系式

YCbCr to RGB24

Y = 0.299R + 0.587G + 0.114B

Cb = 0.568(B-Y) + 128 = –0.172R – 0.339G + 0.511B + 128

Cr = 0.713(R-Y) + 128 = 0.511R – 0.428G – 0.083B + 128

Y = ((77 * R + 150 * G + 29 * B) >> 8);

Cb = ((-43 * R - 85 * G + 128 * B) >> 8) + 128;

Cr = ((128 * R - 107 * G - 21 * B) >> 8) + 128;

RGB24 to YCbCr

R = Y + 1.371(Cr – 128)

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

B = Y + 1.732(Cb – 128)

R = Y + (351*(Cr – 128)) >> 8

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

B = Y + (443*(Cb – 128)) >> 8

# 4. 选择输出方案

## *4.1 后端有 ISP*

如果后端芯片有 ISP（完整 ISP 和 YCbCr ISP),ISP 可做图像缩放。因此，OV7675 只要输出 VGA，通过 ISP 来做处理得到手机需要的各种尺寸。

## *4.2 后端没有 ISP*

如果后端芯片没有对图像做缩放的功能，那就要让 OV7675 输出 LCD 所需尺寸。例如，LCD 尺寸是 176x220，就输出这种尺寸的数据。但是由于 OV7675 没有 scale 功能，就需要通过裁剪来输出给后端所需要的尺寸，这样每种尺寸的视场角是不一样的，尺寸越小，则视场角越小。

# 5. 帧率调整

对 60Hz 灯源，建议用 30fps 和 15fps。对 50Hz 灯源, 建议用 25fps 和 14.3fps。夜间模式帧率应更低，后面还会讨论。PCLK 可分频和数字倍频，建议用分频，不用数字倍频 (register 0x11[7]). 后面会有推荐帧率的设置。

上面推荐帧率的参数见下表。

## *5.1 24Mhz 输入时钟，帧率的调整*

### 30 fps, PCLK = 24Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x00);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 15 fps, PCLK = 12Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x00);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 25fps, PCLK = 24Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x66);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 14.3fps, PCLK = 12Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x1a);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

## *5.2 26Mhz 输入时钟帧率的调整*

### 30 fps, PCLK = 26Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x2b);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 15 fps, PCLK = 13Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x2b);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 25fps, PCLK = 26Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x99);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 14.3fps, PCLK = 13Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x46);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

## *5.3 13Mhz 输入时钟，帧率的调整*

### 30 fps, PCLK = 26Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x2b);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 15 fps, PCLK = 13Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x2b);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 25fps, PCLK = 26Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x99);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

### 14.3fps, PCLK = 13Mhz

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x2a, 0x00);
write_cmos_sensor(0x2b, 0x00);
write_cmos_sensor(0x92, 0x46);
write_cmos_sensor(0x93, 0x00);
write_cmos_sensor(0x3b, 0x0a);
```

## 6. 夜间模式

夜间模式有两种设置。一种是固定帧率，如：3.75fps。另一种是 30fps 到 3.75fps。外界环境亮时，可提高到 30fps 。外界环境暗时，可降到 3.75fps .

### *6.1 固定帧率夜间模式*

### For 24Mhz/26Mhz Clock Input

3.75fps night mode for 60Hz light environment

```
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x03);
```

write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0x0a);

3.125fps night mode for 50Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x03);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0x0a);

### For 13Mhz Clock Input

3.75fps night mode for 60Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0x0a);

3.125fps night mode for 50Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0x0a);

## *6.2 帧率自动调整夜间模式*

### For 24Mhz/26Mhz Clock Input

30fps ~ 3.75fps night mode for 60Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x80);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0x8a);
write_cmos_sensor(0xcf, 0x8c);

15fps ~ 3.75fps night mode for 60Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0xea);
write_cmos_sensor(0xcf, 0x84);

25fps ~ 3.125fps night mode for 50Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x80);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0x8a);

write_cmos_sensor(0xcf, 0x8c);

14.3fps ~ 3.6fps night mode for 50Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x3b, 0xea);
write_cmos_sensor(0xcf, 0x84);

## For 13Mhz Clock Input

30fps ~ 3.75fps night mode for 60Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x3b, 0x8a);
write_cmos_sensor(0xcf, 0x8c);

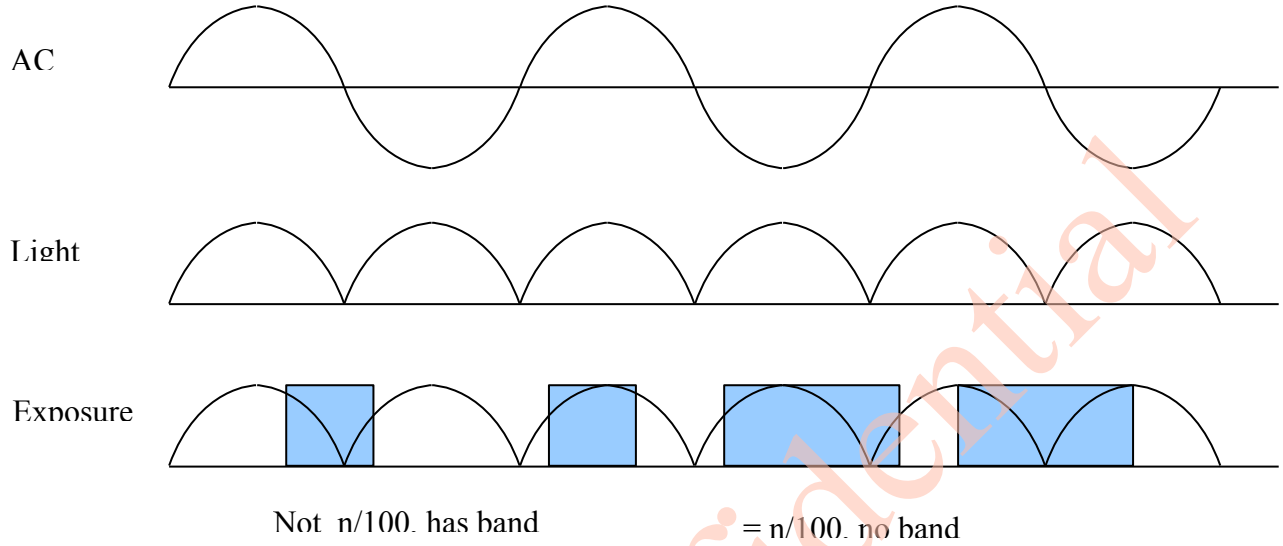
15fps ~  3.75fps night mode for 60Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x3b, 0xea);
write_cmos_sensor(0xcf, 0x84);

25fps ~ 3.125fps night mode for 50Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x00);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x3b, 0x8a);
write_cmos_sensor(0xcf, 0x8c);

14.3fps ~ 3.6fps night mode for 50Hz light environment
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x11, 0x01);
write_cmos_sensor(0x6b, 0x4a);
write_cmos_sensor(0x3b, 0xea);
write_cmos_sensor(0xcf, 0x84);

# 7. 消除 Light Band（灯光条纹）

## *7.1 Light Band*




办公室灯光的强度不是均匀的，它随交流电（AC）频率变化。例如：如果交流电频率是 50Hz,灯光强度变化的频率就是 100Hz.

## *7.2 消除 Light band（灯光条纹）*

为消除 Light band，让曝光时间为 n/100 (n/120 for 60Hz)seconds 。banding filter 的值就是告诉 OV7675 1/100 (1/120 for 60Hz) seconds 相当于多少行。

## *7.3 根据实际条件选择 Banding Filter*

根据手机实际条件选择 banding filter 的值。灯光频率选择表格指出这个区域是用 50Hz 灯光还是 60HZ 灯光，选择了该区域的灯光信息，就得到了灯光频率相关信息。不同灯光频率会有不同帧率。因此，50Hz 灯光和 60Hz 灯光都有对应的优化推荐帧率。

### *24Mhz 时钟，Banding Filter 设置*

30fps for 60Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0x98); //50Hz banding filter
write_cmos_sensor(0x9e, 0x7f); //60Hz banding filter
write_cmos_sensor(0xa5, 0x02); //3 step for 50hz
write_cmos_sensor(0xab, 0x03); //4 step for 60hz
write_cmos_sensor(0x3b, 0x02); //Select 60Hz banding filter

15fps for 60Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0x4c); //50Hz banding filter
write_cmos_sensor(0x9e, 0x3f); //60Hz banding filter
write_cmos_sensor(0xa5, 0x05); //6 step for 50hz
write_cmos_sensor(0xab, 0x07); //8 step for 60hz
write_cmos_sensor(0x3b, 0x02); //Select 60Hz banding filter

25fps for 50Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0x98); //50Hz banding filter
write_cmos_sensor(0x9e, 0x7f); //60Hz banding filter
write_cmos_sensor(0xa5, 0x03); //4 step for 50hz
write_cmos_sensor(0xab, 0x03); //4 step for 60hz
write_cmos_sensor(0x3b, 0x0a); //Select 50Hz banding filter

14.3fps for 50Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0x4c); //50Hz banding filter
write_cmos_sensor(0x9e, 0x3f); //60Hz banding filter
write_cmos_sensor(0xa5, 0x06); //7 step for 50hz
write_cmos_sensor(0xa5, 0x07); //8 step for 60hz
write_cmos_sensor(0x3b, 0x0a); //Select 50Hz banding filter

### *13Mhz/26Mhz 时钟，Banding Filter 设置*

30fps for 60Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0xa5); //50Hz banding filter
write_cmos_sensor(0x9e, 0x89); //60Hz banding filter
write_cmos_sensor(0xa5, 0x02); //3 step for 50hz
write_cmos_sensor(0xab, 0x03); //4 step for 60hz
write_cmos_sensor(0x3b, 0x02); //Select 60Hz banding filter

15fps for 60Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0x52); //50Hz banding filter
write_cmos_sensor(0x9e, 0x44); //60Hz banding filter
write_cmos_sensor(0xa5, 0x06); //7 step for 50hz
write_cmos_sensor(0xab, 0x07); //8 step for 60hz
write_cmos_sensor(0x3b, 0x02); //Select 60Hz banding filter

25fps for 50Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0xa5); //50Hz banding filter
write_cmos_sensor(0x9e, 0x89); //60Hz banding filter
write_cmos_sensor(0xa5, 0x02); //3 step for 50hz
write_cmos_sensor(0xab, 0x03); //4 step for 60hz
write_cmos_sensor(0x3b, 0x0a); //Select 50Hz banding filter

14.3fps for 50Hz light frequency
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //banding filter enable
write_cmos_sensor(0x9d, 0x52); //50Hz banding filter
write_cmos_sensor(0x9e, 0x44); //60Hz banding filter
write_cmos_sensor(0xa5, 0x06); //7 step for 50hz
write_cmos_sensor(0xa5, 0x07); //8 step for 60hz
write_cmos_sensor(0x3b, 0x0a); //Select 50Hz banding filter

## *7.4 通过自动灯光频率检测，选择 Banding Filter 值*

OV7675 不支持自动灯光的检测。

## *7.5 Light Band 不能消除时*

一般 Light Band 是通过 banding filter 消除。
有些特殊环境，如阳光和办公室灯光的混合光，图片就会有条纹，这个 Light Band 是不

能消除的。因曝光时间小于 1/100 秒（ 50hz ）和 1/120 秒（60hz).

这种条件下的 Light Band，不仅仅是 OV7675，对所有 CMOS sensors 都不能消除，也没有办法消除。

# 8.白平衡

## *8.1 简单白平衡*

OV7675 支持简单白平衡。
简单白平衡就是一种设想的“灰阶世界”。世界的平均颜色是灰白的，实际上大部分环境是这样的。

简单白平衡的优点
简单白平衡不依靠镜头。简单白平衡时，一般的设置可用于所有不同镜头的模组。

简单白平衡的缺点
在不是“灰阶世界”的环境下，不准确。如背景有一块巨大的红、蓝或绿等，就不准确。如果 camera 对准单一颜色，如红、蓝或绿，简单白平衡会产生单一颜色灰阶的图像。
Settings
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //AWB on
write_cmos_sensor(0x6f, 0x9f); // Simple AWB

## *8.2 高级白平衡*

高级白平衡是使用色域信息来检测白色区域，做白平衡。

高级白平衡优点
颜色比简单白平衡更准确。即使背景是单一颜色， camera 也不会做出单一色彩灰阶图像。

高级白平衡缺点
高级白平衡的设置和镜头有关，对于不同类型新镜头，要调试不同参数。必需由 OmniVision FAE 在光学实验室用一些光学设备如：灯箱、色卡等，调试参数。

参数
联系 OmniVision 本地 技术支持工程师。

**注：为客户获得很好的影象效果，OmniVision 推荐使用高级白平衡。**

## 9. 死伤点的校正

死伤点包括死点和伤点。

死点包括白点和黑点。白点指无论外界图片光线亮暗总是白的点。 黑点指无论外界图片光线亮暗总是黑的点

伤点会随外界光线变化，但和正常点不一样，白伤点就是比正常像素点要亮，黑伤点就是比正常像素点要暗，但没全黑。

OV7675 内部有死伤点的校正功能。OV7675 如果输出 YCbCr, RGB565, Processed raw RGB，死伤点的校正功能可处理死伤点，输出 Bayer raw RGB 时，死伤点的校正功能不能用，必须用后端芯片的死伤点的校正功能。

请注意后端芯片的死伤点的校正功能，有些后端芯片的死伤点校正功能不能完全校正 OV7675 的死伤点。

## 10. BLC（暗电流补偿）

暗电流补偿功能（BLC）是使暗光环境下图片颜色更精确。OV7675 有自动 BLC 功能，一般总是打开。

## 11. 录像模式

录像模式要求高帧率，通常 15fps。 对录像没有夜间模式。

## 12. 数字缩放

OV7675 不支持数字缩放
后端芯片一般可支持多种等级数字缩放。

# 13. OV7675 功能

## *13.1 环境模式*

Auto
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //AWB on
write_cmos_sensor(0x3b, 0x0a);
write_cmos_sensor(0x2d, 0x00);
write_cmos_sensor(0x2e, 0x00);

Sunny
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe5); //AWB off
write_cmos_sensor(0x01, 0x5a);
write_cmos_sensor(0x02, 0x5c);
write_cmos_sensor(0x6a, 0x42);
write_cmos_sensor(0x3b, 0x0a);
write_cmos_sensor(0x2d, 0x00);
write_cmos_sensor(0x2e, 0x00);

Cloudy
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe5); //AWB off
write_cmos_sensor(0x01, 0x58);
write_cmos_sensor(0x02, 0x60);
write_cmos_sensor(0x6a, 0x40);
write_cmos_sensor(0x3b, 0x0a);
write_cmos_sensor(0x2d, 0x00);
write_cmos_sensor(0x2e, 0x00);

Office
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe5); //AWB off
write_cmos_sensor(0x01, 0x84);
write_cmos_sensor(0x02, 0x4c);
write_cmos_sensor(0x6a, 0x40);
write_cmos_sensor(0x3b, 0x0a);
write_cmos_sensor(0x2d, 0x00);
write_cmos_sensor(0x2e, 0x00);

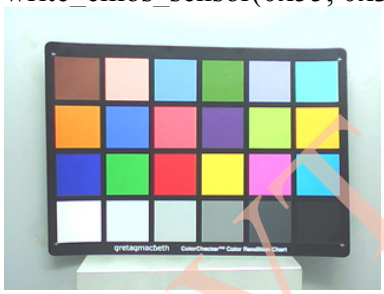
Home
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe5); //AWB off
write_cmos_sensor(0x01, 0x96);
write_cmos_sensor(0x02, 0x40);
write_cmos_sensor(0x6a, 0x4a);
write_cmos_sensor(0x3b, 0x0a);
write_cmos_sensor(0x2d, 0x00);
write_cmos_sensor(0x2e, 0x00);

Night
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x13, 0xe7); //AWB on
write_cmos_sensor(0x3b, 0x0a);
write_cmos_sensor(0x2d, 0x00);
write_cmos_sensor(0x2e, 0x00);
write_cmos_sensor(0x3b, 0xea);

## *13.2 曝光补偿*

OV7675 曝光的值可调节。高曝光值会使图片更亮。

Brightness +2
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x55, 0x30);



Brightness +1
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x55, 0x18);

Brightness 0
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x55, 0x00);

Brightness -1
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x55, 0x98);

Brightness -2
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x55, 0xb0);

### *13.3 对比度*

OV7675 对比度可调节。高对比度会使图片更锐利，但会减少动态范围。

Contrast +2
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x56, 0x60);

Contrast +1
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x56, 0x50);

Contrast 0
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x56, 0x40);

Contrast -1
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x56, 0x38);

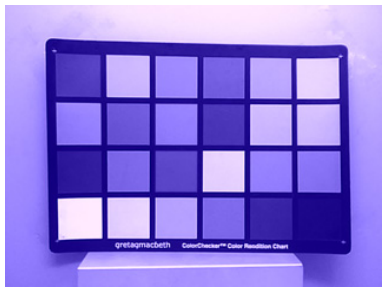
Contrast -2
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x56, 0x30);

## *13.4 特效*

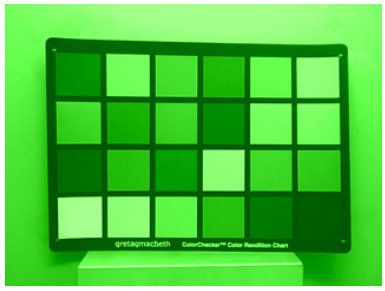
OV7675 支持一些特效，如黑/白、负片、棕褐色、蓝色、红色、绿色等，如果要其它特效可让后端芯片支持。

Antique
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x14);
write_cmos_sensor(0x67, 0xa0);
write_cmos_sensor(0x68, 0x40);

Bluish
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x14);
write_cmos_sensor(0x67, 0x80);
write_cmos_sensor(0x68, 0xc0);



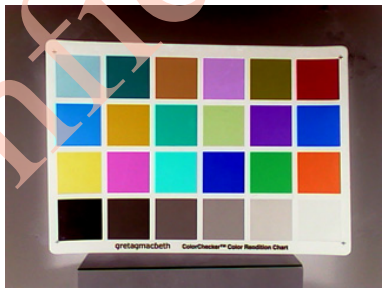
Greenish
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x14);
write_cmos_sensor(0x67, 0x40);
write_cmos_sensor(0x68, 0x40);



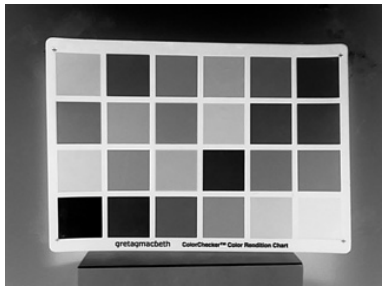
Redish
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x14);
write_cmos_sensor(0x67, 0xc0);
write_cmos_sensor(0x68, 0x80);

B&W
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x14);
write_cmos_sensor(0x67, 0x80);
write_cmos_sensor(0x68, 0x80);

Negative
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x24);
write_cmos_sensor(0x67, 0x80);
write_cmos_sensor(0x68, 0x80);



B&W negative
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x34);
write_cmos_sensor(0x67, 0x80);
write_cmos_sensor(0x68, 0x80);



Normal
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x3a, 0x04);
write_cmos_sensor(0x67, 0xc0);
write_cmos_sensor(0x68, 0x80);

# 14. 镜头处理

## *14.1 镜头失光*

镜头失光是图片四个角比中心暗。这由镜头导致的。可打 OV7675 镜头补偿功能，去提高四个角的亮度，让整个图片看起来一样亮。

## *14.2 暗角*

一些镜头会有暗角。暗角指图片四个角看起来是黑的。不可能用镜头补偿功能来校正暗角。有暗角的模组是不合格的，不能用。

## *14.3 分辨率*

模组分辨率依靠于镜头设计、焦距调整和 sensor（图像传感器）分辨率等。在模组装配中调焦是非常重要的。OV7675 对焦距离在 40~50cm 。视野的深度从 20~25cm 到无穷远。如果要检查模组的分辨率，解析度卡应放在 40~50 cm 处。

## *14.4 光学对比度*

镜头的光学对比度对图片质量是非常重要的。如果镜头的光学对比度不好，图片会显得模糊。虽然可通过调节 sensor 对比度，提高图片的锐度，但高对比度会使图片丢掉黑色区域的细节。

## *14.5 保护玻璃*

保护玻璃是便宜的光学器件，但对图片影响很大，保护玻璃应是两边都有 AR coating（增透膜）的光学玻璃。否则，保护玻璃会导致灵敏度丢失和很大镜头耀斑。

## *14.6 镜头补偿*

每个模组的镜头补偿参数都要调整。请联系 OmniVision 本地技术支持工程师。

# 15. 初始化参数

## *15.1 YCbCr*

```
//Input clock 24Mhz, 25fps
//SCCB_salve_Address = 0x42;
write_cmos_sensor(0x12, 0x80);
//
write_cmos_sensor(0x11, 0x80);
write_cmos_sensor(0x3a, 0x04);
write_cmos_sensor(0x12, 0x00);
```

```
write_cmos_sensor(0x17, 0x13);
write_cmos_sensor(0x18, 0x01);
write_cmos_sensor(0x32, 0xb6);
write_cmos_sensor(0x19, 0x03);
write_cmos_sensor(0x1a, 0x7b);
write_cmos_sensor(0x03, 0x0a);
write_cmos_sensor(0x0c, 0x00);
write_cmos_sensor(0x3e, 0x00);
write_cmos_sensor(0x70, 0x3a);
write_cmos_sensor(0x71, 0x35);
write_cmos_sensor(0x72, 0x11);
write_cmos_sensor(0x73, 0xf0);
write_cmos_sensor(0xa2, 0x02);
//
write_cmos_sensor(0x7a, 0x18);
write_cmos_sensor(0x7b, 0x04);
write_cmos_sensor(0x7c, 0x09);
write_cmos_sensor(0x7d, 0x18);
write_cmos_sensor(0x7e, 0x38);
write_cmos_sensor(0x7f, 0x47);
write_cmos_sensor(0x80, 0x56);
write_cmos_sensor(0x81, 0x66);
write_cmos_sensor(0x82, 0x74);
write_cmos_sensor(0x83, 0x7f);
write_cmos_sensor(0x84, 0x89);
write_cmos_sensor(0x85, 0x9a);
write_cmos_sensor(0x86, 0xA9);
write_cmos_sensor(0x87, 0xC4);
write_cmos_sensor(0x88, 0xDb);
write_cmos_sensor(0x89, 0xEe);
//
write_cmos_sensor(0x13, 0xe0);
write_cmos_sensor(0x01, 0x50);
write_cmos_sensor(0x02, 0x68);
write_cmos_sensor(0x00, 0x00);
write_cmos_sensor(0x10, 0x00);
write_cmos_sensor(0x0d, 0x40);
write_cmos_sensor(0x14, 0x18);
write_cmos_sensor(0xa5, 0x07);
write_cmos_sensor(0xab, 0x08);
write_cmos_sensor(0x24, 0x60);
write_cmos_sensor(0x25, 0x50);
write_cmos_sensor(0x26, 0xe3);
write_cmos_sensor(0x9f, 0x78);
write_cmos_sensor(0xa0, 0x68);
//
write_cmos_sensor(0xa1, 0x03);
write_cmos_sensor(0xa6, 0xd8);
```

```
write_cmos_sensor(0xa7, 0xd8);
write_cmos_sensor(0xa8, 0xf0);
write_cmos_sensor(0xa9, 0x90);
write_cmos_sensor(0xaa, 0x14);
write_cmos_sensor(0x13, 0xe5);
//
write_cmos_sensor(0x0e, 0x61);
write_cmos_sensor(0x0f, 0x4b);
write_cmos_sensor(0x16, 0x02);
write_cmos_sensor(0x1e, 0x07);
write_cmos_sensor(0x21, 0x02);
write_cmos_sensor(0x22, 0x91);
write_cmos_sensor(0x29, 0x07);
write_cmos_sensor(0x33, 0x0b);
write_cmos_sensor(0x35, 0x0b);
write_cmos_sensor(0x37, 0x1d);
write_cmos_sensor(0x38, 0x71);
write_cmos_sensor(0x39, 0x2a);
write_cmos_sensor(0x3c, 0x78);
write_cmos_sensor(0x4d, 0x40);
write_cmos_sensor(0x4e, 0x20);
write_cmos_sensor(0x69, 0x00);
write_cmos_sensor(0x6b, 0x0a);
write_cmos_sensor(0x74, 0x10);
write_cmos_sensor(0x8d, 0x4f);
write_cmos_sensor(0x8e, 0x00);
write_cmos_sensor(0x8f, 0x00);
write_cmos_sensor(0x90, 0x00);
write_cmos_sensor(0x91, 0x00);
write_cmos_sensor(0x92, 0x66);
write_cmos_sensor(0x96, 0x00);
write_cmos_sensor(0x9a, 0x80);
write_cmos_sensor(0xb0, 0x84);
write_cmos_sensor(0xb1, 0x0c);
write_cmos_sensor(0xb2, 0x0e);
write_cmos_sensor(0xb3, 0x82);
write_cmos_sensor(0xb8, 0x0a);
//
write_cmos_sensor(0x43, 0x14);
write_cmos_sensor(0x44, 0xf0);
write_cmos_sensor(0x45, 0x41);
write_cmos_sensor(0x46, 0x66);
write_cmos_sensor(0x47, 0x2a);
write_cmos_sensor(0x48, 0x3e);
write_cmos_sensor(0x59, 0x8d);
write_cmos_sensor(0x5a, 0x8e);
write_cmos_sensor(0x5b, 0x53);
write_cmos_sensor(0x5c, 0x83);
```

```
write_cmos_sensor(0x5d, 0x4f);
write_cmos_sensor(0x5e, 0x0e);
write_cmos_sensor(0x6c, 0x0a);
write_cmos_sensor(0x6d, 0x55);
write_cmos_sensor(0x6e, 0x11);
write_cmos_sensor(0x6f, 0x9e);
//
write_cmos_sensor(0x62, 0x90);
write_cmos_sensor(0x63, 0x30);
write_cmos_sensor(0x64, 0x11);
write_cmos_sensor(0x65, 0x00);
write_cmos_sensor(0x66, 0x05);
write_cmos_sensor(0x94, 0x11);
write_cmos_sensor(0x95, 0x18);
//
write_cmos_sensor(0x6a, 0x40);
write_cmos_sensor(0x01, 0x40);
write_cmos_sensor(0x02, 0x40);
write_cmos_sensor(0x13, 0xe7);
//
write_cmos_sensor(0x4f, 0x80);
write_cmos_sensor(0x50, 0x80);
write_cmos_sensor(0x51, 0x00);
write_cmos_sensor(0x52, 0x22);
write_cmos_sensor(0x53, 0x5e);
write_cmos_sensor(0x54, 0x80);
write_cmos_sensor(0x58, 0x9e);
//
write_cmos_sensor(0x41, 0x08);
write_cmos_sensor(0x3f, 0x00);
write_cmos_sensor(0x75, 0x03);
write_cmos_sensor(0x76, 0xe1);
write_cmos_sensor(0x4c, 0x00);
write_cmos_sensor(0x77, 0x00);
write_cmos_sensor(0x3d, 0xc2);
write_cmos_sensor(0x4b, 0x09);
write_cmos_sensor(0xc9, 0x60);
write_cmos_sensor(0x41, 0x38);
write_cmos_sensor(0x56, 0x40);
//
write_cmos_sensor(0x34, 0x11);
write_cmos_sensor(0x3b, 0x0a);
write_cmos_sensor(0xa4, 0x88);
write_cmos_sensor(0x96, 0x00);
write_cmos_sensor(0x97, 0x30);
write_cmos_sensor(0x98, 0x20);
write_cmos_sensor(0x99, 0x30);
write_cmos_sensor(0x9a, 0x84);
```

```
write_cmos_sensor(0x9b, 0x29);
write_cmos_sensor(0x9c, 0x03);
//
write_cmos_sensor(0x9d, 0x98);
write_cmos_sensor(0x9e, 0x3f);
write_cmos_sensor(0x78, 0x04);
//
write_cmos_sensor(0x79, 0x01);
write_cmos_sensor(0xc8, 0xf0);
write_cmos_sensor(0x79, 0x0f);
write_cmos_sensor(0xc8, 0x00);
write_cmos_sensor(0x79, 0x10);
write_cmos_sensor(0xc8, 0x7e);
write_cmos_sensor(0x79, 0x0a);
write_cmos_sensor(0xc8, 0x80);
write_cmos_sensor(0x79, 0x0b);
write_cmos_sensor(0xc8, 0x01);
write_cmos_sensor(0x79, 0x0c);
write_cmos_sensor(0xc8, 0x0f);
write_cmos_sensor(0x79, 0x0d);
write_cmos_sensor(0xc8, 0x20);
write_cmos_sensor(0x79, 0x09);
write_cmos_sensor(0xc8, 0x80);
write_cmos_sensor(0x79, 0x02);
write_cmos_sensor(0xc8, 0xc0);
write_cmos_sensor(0x79, 0x03);
write_cmos_sensor(0xc8, 0x40);
write_cmos_sensor(0x79, 0x05);
write_cmos_sensor(0xc8, 0x30);
write_cmos_sensor(0x79, 0x26);
write_cmos_sensor(0x2d, 0x00);
write_cmos_sensor(0x2e, 0x00);
```

注：针对不同的 DOVDD 电压，需要设置不同的寄存器值。

1.如果 0xC2[2] = 1 (默认值)，

对 DOVDD = 2.8V，则 0xB8[6:3] 设为 4'b0001

对 DOVDD = 1.8V，则 0xB8[6:3] 设为 4'b0010

2.如果 0xC2[2] = 0

对 DOVDD = 2.8V，则 0xDA[3:0] 设为 4'b0001

对 DOVDD = 1.8V，则 0xDA[3:0] 设为 4'b0010


以上初始化参数已经针对 OmniVision 认证模组进行优化，一般不需作调整即可使用。在 OV7670 兼容模式下，其中一些效果已经固定，包括颜色，镜头补偿，白平衡，对比度，锐度，降噪等。当然这些效果在 OV7675 模式也是可以调整的，如果有对图象效果上的特别要求，请联系 OmniVision 本地技术支持工程师。

### *15.2 RGB raw*

联系 OmniVision 本地技术支持工程师。

### *15.3 RGB565*

联系 OmniVision 本地技术支持工程师。

Revision History

**Rev1.00**

Initial release

**Rev1.01**

更新 4.夜间模式

**Rev1.02**

更新 OmniVision 认证模组列表(采用 OV7675)，增加 OV7670 兼容模式