

Micron sensor 驱动与调试小结

苏豫

MSN: suyuwang3@hotmail.com

目录:

前言

驱动篇:

- 1、Micron sensor ISP 的原理图
- 2、sensor 的原理框架
- 3、Sensor 的初始化步骤
- 4、Preview 时候的 sensor 设置
- 5、Capture 时候的 sensor 设置
- 6、工频干扰的调试
- 7、亮度以及夜景模式

A
B

调试篇:

- 1、清晰度的测试
- 2、灰阶重现
- 3、画面的均匀性以及暗脚补偿
- 4、畸变
- 5、白平衡的调试

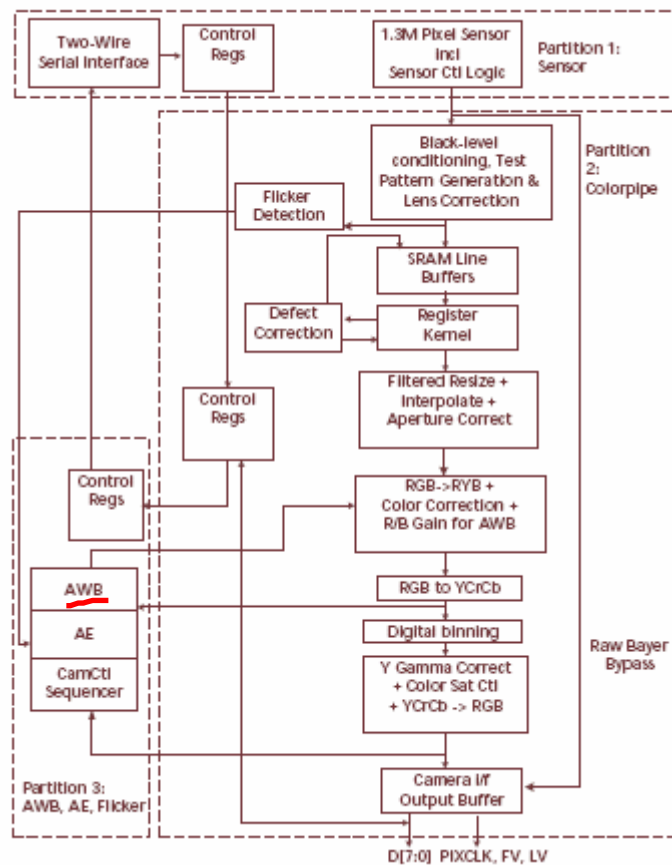
前言

Micron sensor 是我们公司所用最多的图像传感器,也是目前市场上评价很高的主流 sensor 产品。写这篇文章的目的在于让后继调试 sensor 者对 sensor 的调试有初步的思路和对 micron sensor 的一些特性有一定的了解,希望以后的调试工作能够有少走一些弯路。具体的 sensor 的工作原理和更深入的图像工程方面的

知识,可以参看各个 sensor 的 datasheet 和上网查找一些关于 camera 的测试资料。

驱动篇：

Micron sensor ISP 的原理图:



下图是 **sensor** 的功能框架构图:

Figure 1: Functional Block Diagram

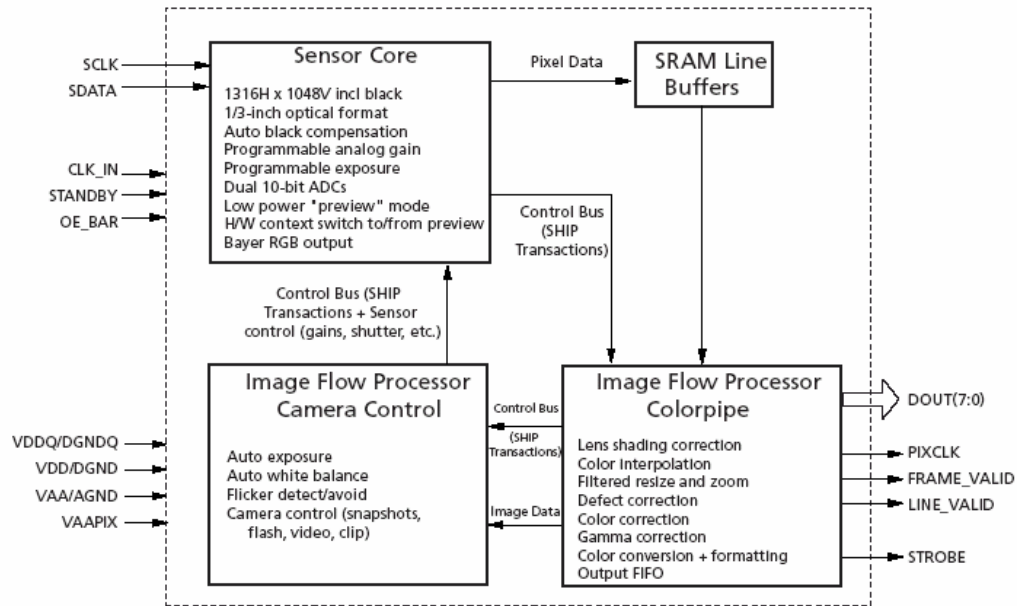
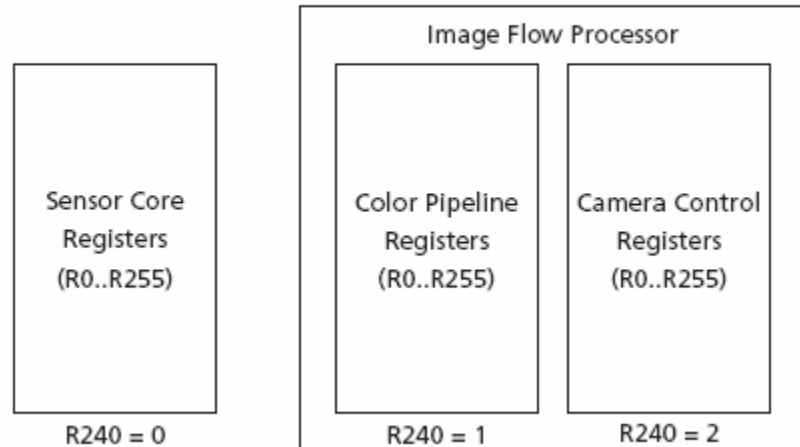


Figure 2: Internal Registers



① Sensor Core register 是实际上控制 sensor 的 register.是直接控制 sensor 的寄存器(对应的是 sensor 寄存器的 page 0)。

② Image Flow Processor 里的 register 主要是一些控制 sensor 的算法的寄存器。其中 color Pipeline 主要是对输出数据和信号的一些控制。比如 Base configuration, lens shading, resize, output format

(page 1)

③ Camera control集中了对sensor core的控制算法, 控制sensor core的工作都是在这个寄存器组中完成。(page 2) 比如AE, AWB, Flicker, Camera control

sequencer。

Sensor 的初始化步骤:

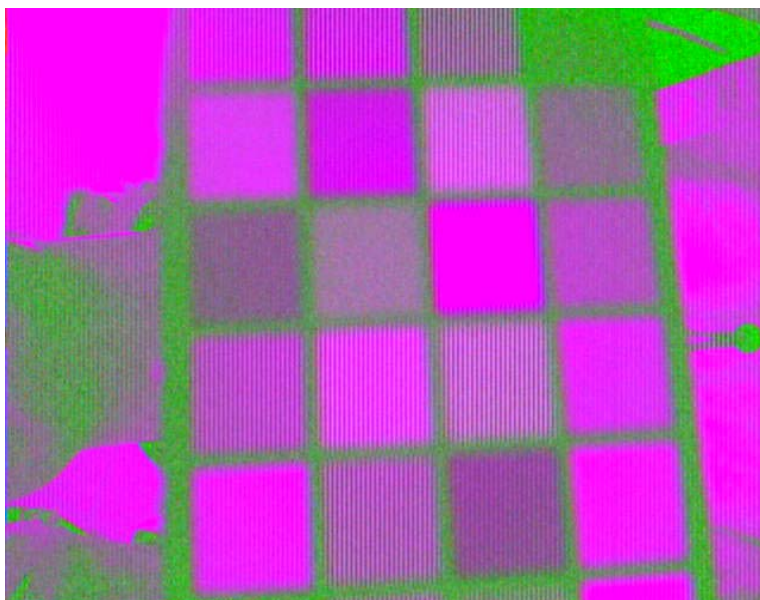
一般 sensor 的初始化通常包含以下几个步骤:

- 1、 sensor 的上电。Micron sensor 的电源分为数字电源，模拟电源和 IO 电压。这三个电源并没有严格的先后上电顺序，可以在代码中同时打开。
- 2、 对 sensor 输出 MCLK，配置对 PCLK 采样输出频率，这是能否正常接收 sensor 数据的关键。
- 3、 配置 V，H 同步信号的输出极性，如果极性配置不对，将造成图象不能正常采集，自然显示混乱。
- 4、 硬件的 reset。 Micron sensor 的 reset 为低 reset，并且至少持续 1US，
- 5、 软件的 reset。既然为软件的 reset，那就必须要求 BB 或者多媒体 MAP 能够对 sensor 进行写寄存器。也就是要保证 IIC 能够正常地写数据进入 sensor ISP,这点是保证软件能够进行调试的基础。软件 reset 通常根据 sensor 的不同会有所变化，如 mt9d111 内带一个 MCU，所以在 reset 的时候要对 MCU 同时进行 reset。而 mt9m11 就没有带 MCU。注意硬件 reset 后要保留一些时间才能使用 IIC 总线，通常在 10 个 US 以上。
- 6、 Micron sensor mtd9111 系列的带 ISP 的 2M sensor 在 ISP 中默认了一组寄存器，能够在 reset 后不用 IIC 写任何寄存器就能输出图象，这个时候 sensor 的 input clock 是 output clock 的两倍，前期可以用这个方法验证硬件和软件供电，复位等是否正确，当后端接受的图象 engine 只能用 mclk 来同步工作时候，必须要正确配置接收的采样频率，否

则不能得出正确的图象。



上图就是采样频率不匹配的现象。请注意与 YUV，RGB 顺序配置错误的现象有什么区别。下图是顺序倒置：



- 7、 写入 micron 工程师给的初始化寄存器，并配置输出频率和输出图象的分辨率。
- 8、 读取 sensor 的版本号，如果与我们所用产品的 version 一致，就代表初始化工作正确完成。

初始化 sensor 的道理很简单，而且如果平台比较成熟，有可能一次性就能正确的初始化，也有可能花很多时间去查问题，特别是如果你遇到了 iic 写给 sensor 的时候出现不稳定的现象或者 sensor 接收到 iic 命令和数据，却不按正常地输出，那就比较麻烦了，不过 2m sensor 的 ISP 既然已经带了

MCU，不妨把它当成一个应用处理器去维护，在写某些改变 sensor 内部工作状态的寄存器时要注意延时。有些时候对一个寄存器可以多次写入保证其能正常地工作，这是一个还没有想出原因的经验。按照以上的流程检查应该能够准确定位 80% 以上的问题。

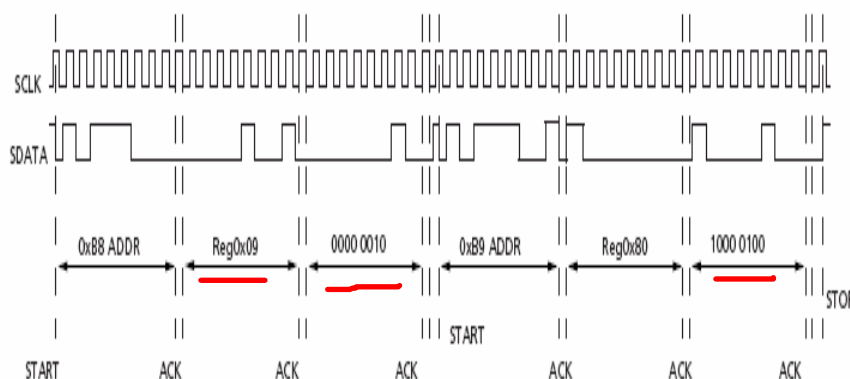
IIC 总线协议

I²C 串行总线有两根信号线：一根双向的数据线 SDA；另一根是时钟线 SCL。所有接到 I²C 总线上的设备的串行数据都接到总线的 SDA 线，各设备的时钟线 SCL 接到总线的 SCL。在 Vienna 平台上，I²C 连了两个器件，分别是 sensor 和 audio codec，，主控设备（MAP）发送不同的器件 ID 与两个设备进行数据传输。

在 I²C 总线传输过程中，将两种特定的情况定义为开始和停止条件，当 SCL 保持“高”，SDA 由“高”变为“低”时为开始条件；SCL 保持“高”，SDA 由“低”变为“高”是为停止条件。开始和停止条件由主控器产生。使用硬件接口可以很容易地检测开始和停止条件，没有这种接口的微机必须以每时钟周期至少两次对 SDA 取样以使检测这种变化。

下图对就是 IIC 的一次写操作，具体的 IIC 协议很容易在网站上可以找到，micron 的 datasheet 也能找到一些介绍。

Figure 24: Timing Diagram Showing a Write to Reg0x09 with the Value 0x0284



Preview 时候的 sensor 设置：

在 preview 的时候为了得到更高的帧率，通常采用低分辨率的输出，也

就是长宽都只有最高分辨率的一半，本来能够设置更小的分辨率输出，这个时候 sensor 的输出是间隔输出，并不是对象素采样后的均匀输出。但是由于多媒体芯片 corelogic 的无法接收不规则的 pclk 的原因，只有作罢。而有些平台的 camera interface 就不存在这个问题。从而能直接输出屏幕大小的图象，减少后端处理的繁琐和节约为 preview 所开的 buffer 大小。

Capture 时候的 sensor 设置：

Capture 为了获得更大的分辨率和更好的图象质量，所以必须采用高分辨率的输出，那么在切换到 capture 的时候就需要对 sensor 进行一组寄存器设置，micron sensor 为用户提供了两个相对独立的 context，能够存两个寄存器组，默认的设置是将 preview 用 context A, capture 用 context B, context A 通常是 low power mode，而 context B 用的是 full power mode，由于 corelogic 只能用 mclk 去同步采样，这样它便只能接受规则的 pclk，不然采样就会有问✱题，那么这样就要求 context A 与 context B 的都为 full power mode，这样就能保证无论 preview 还是 capture，sensor 都能以已固定的频率的输出 pclk，后端 MAP 就能正常地接收图象信息，不至于出现花屏和颜色不对。由于这样的解决办法并不是 micron 推荐的方案，而是自己想出的弥补方法。所以 Vienna 平台上至少一半以上的问题都是由 corelogic 的这个缺陷，以及为了弥补这个缺陷所修改的 sensor 输出引起的，以后如果要换多媒体应用处理器，请考虑到 MAP 是否支持接受不规则的 pclk。目前 Vienna 和高通平台都不支持变化的 PCLK，而 Vision 平台是支持的。



lower power 模式下的输出



full power 模式下的输出

证明 corelogic 不能接受 lower power mode 的 sensor 输出。

由于输出的时钟频率 preview 与 capture 固定，CMOS sensor 暴光原理是行暴光，暴光时间等于行暴光时间， $\text{line time} = \text{hsync time} + \text{hblank time}$ 。130 万象素的 sensor 为例，preview 的时候输出 VGA， $\text{hsync time} = 640 * k$ ，K 为 shutter width(快门时间)。而 capture 的时候 $\text{hsync time} = 1280 * k$ ，在 k 不变的情况下，hsync time 发生了巨大的变化，这样暴光时间也发生了巨大的变化，现象是拍照的时候的图片明显过曝。不过 micron 给出了一个可以改变 shutter width 的寄存器，通过改变这个寄存器能够调整综合暴光时间，这样就能解决这个问题，在老化测试的时候经常出现暴光不对，就是因为这个寄存器没写入或写入后 sensor 没有反映造成的，由于 preview->capture->preview 中间的转化值都是用软件来实时计算出来的，所以无论是 iic 读或者写，还是环境亮度引起 sensor 亮度计算错误，都会影响到暴光，维护这段代码的时候要特别小心。如果遇到拍照的时候与 preview 的图像质量差别很大，请从这段代码开始查。

```
// sensor 在进行模式切换  
IIC_Write16bit(0xf0, 0x0001);  
IIC_Write16bit(0xC6, 0xA104);  
//判断模式切换是否成功  
dataTemp=IIC_Read16bit(0xc8);
```



```

if(g_nightmode)
{
    waittime=1000;
}
else
{
    waittime=50;
}

```

//判断模式切换是否成功，请注意这个不一定会成功。

```

while (dataTemp != 7)
{

    dataTemp = IIC_Read16bit(0xC8);
    WaitTime_ms(1);
    i++;

    if(i>waittime)
    {
        AMOIT(" the sensor change mega mode fail!----\n");
        break;
    }
}

AMOIT1("-----i=%d----\n",i);

```

//以下这段就是在进行暴光控制

```

IIC_Write16bit(0xf0, 0x0000);
dataTemp = IIC_Read16bit(0x09);
gCurbrightness1 = dataTemp;
IIC_Read16bit(0x09);
WaitTime_ms(10);

```

```
IIC_Write16bit(0x09, dataTemp*2/7);  
IIC_Write16bit(0xf0, 0x0001);  
IIC_Write16bit(0xc6, 0x2225);  
dataTemp = IIC_Read16bit (0xc8);  
gCurbrightness2 = dataTemp;  
WaitTime_ms(10);  
IIC_Write16bit(0xf0, 0x0000);  
IIC_Write16bit(0x65, 0xB000); // CLOCK_ENABLING  
IIC_Write16bit(0x65, 0xE000); // CLOCK_ENABLING  
  
WaitTime_ms(600); // Wait 1 frame time
```



preview 的时候输出的图象。



capture 下来的图象，可以看见明显的过暴现象。

根据经验并非模式切换失败就一定不能正常输出百万像素，而不能正常输出多半就是模式切换失败。模式切换的时间与成功率与帧率有一定的关系，一般说来，帧率越快，时间越短，成功率越高。再从 capture->preview 的时候也必须设置暴光值，以保证图像不会突然变暗，如果发现拍照越来越暗，多半就是返回 preview 的时候设置失败。

注意：模式切换的时候，用示波器可以看到 sensor 在做切换的时候会出现突然拉低 VSYNC 信号，形成一个较长的消隐（blank）信号，然后输出另外一个 MODE 的信号，有时候不稳定的现象就是这个 blank 信号过长，特别是发生在低帧率的情况下，会使后端 ISP 或者 DSP 无法采集到数据，没有办法产生拍照的中断，造成 task 被挂起（进入 idle task）或者死机重启（被狗咬）的现象，具体原因和对策可以根据不同平台的实现方法去分析解决。

到了这里 sensor 的两个基本状态的调试就算基本完成了，后期就是对这段代码的维护工作。

工频干扰：

如果手机出现以如下图的这种水波纹就是工频干扰。工频干扰是由于室内日光灯闪烁造成的。CMOS 与 CCD 两种不同的工艺制造出来的 sensor 工频干扰现象是不一样的，这是由暴光的方式不同造成的。

CMOS 是行暴光，也就是在每行暴光时间决定了画面的亮度，举例：一个 50HZ 的光源，电压曲线为正弦曲线，那能量曲线定性分析可以认为是取了绝对值的电压曲线。那就是能量做 1/100 秒的周期变化。那就要求暴光的时间必须是 1/100 秒的整数倍。如果没有把暴光时间调整到 1/100 秒的整数倍，就有可能会有每行的暴光值不一样，造成同一个 image 上有水波纹现象。CCD 是整帧同时暴光，所以，工频干扰表现的就是图像有轻微的闪烁。产生的原理与 CMOS sensor 的原理相似。

如果有发现这样的问题，可先计算出暴光时间，再在这个基础上进行微调。相信很快就能调到没有工频干扰。

Micron 有个寄存器能够调整暴光，以达到消除工频干扰的目的。

算法： $\text{line time} * 0x58(\text{page 2}) / \text{PCLK} = N / 100$ (用这个公式算出来的值还要进行微调试，reg 0x58 是 mt9m111 的寄存器，不同的 sensor 是不一样的，但一定能找到一个类似的寄存器，N 是自然数，datasheet 里面没有介绍，呵呵，不过如果你了解 cmos 的暴光原理，相信很容易明白的)。

亮度以及夜景模式：

相信现在大家都知道图象的亮度与暴光时间相关，所以为了让暗处的图片能够清晰地显示必须增加 sensor 对暗处图象的暴光时间，也就是 line time 会设置得比普通模式的时候要大多，这样能使 CMOS sensor 拥有更多的暴光时间，从而提高亮度。



上图为没有采用夜景模式的照片，下图为使用夜景模式的照片



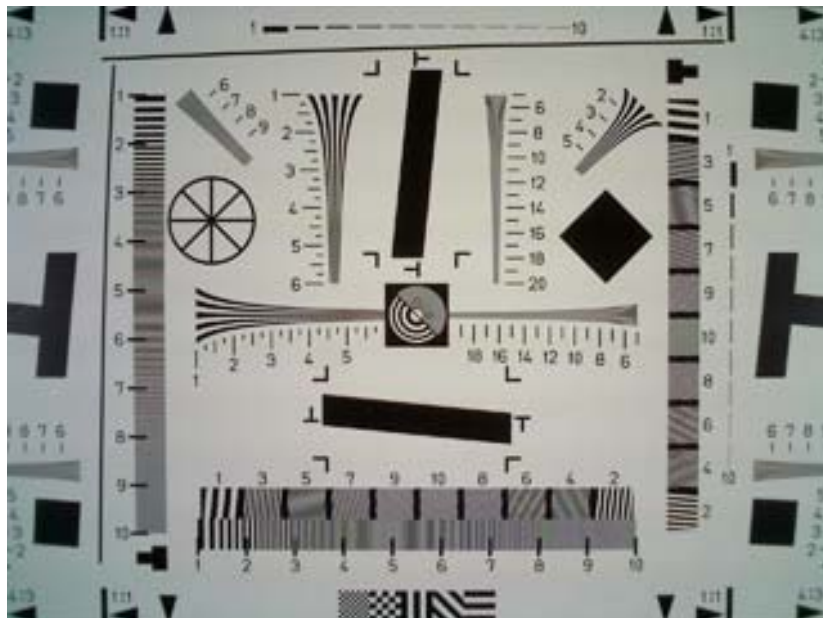
我们有两种方法来控制图像亮度，一种是使用 AE target，一种是加大灰度增益。我们使用的是 AE target 方式，这样的图片色彩更逼真。用这种办法会影响到 frame rate，当帧率达到我们限制的极限的时候，就要用增加模拟增益来做了，这样会同时放大图像噪声。一般不用增加数字放大增益来调整图像亮度。

调试篇

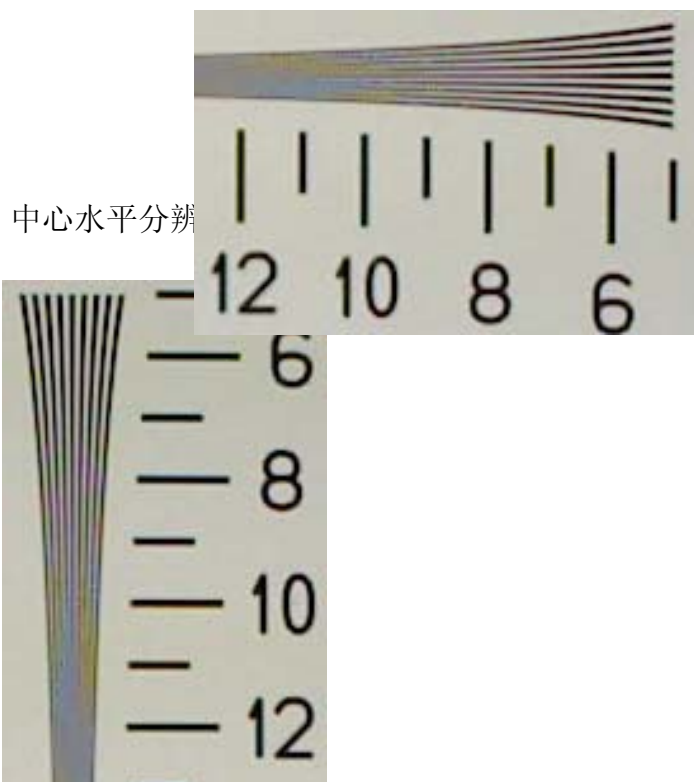
与一般 IC 的驱动不同，sensor 除了要工作起来和稳定性之外，还需要调试图像质量。在这点上，所有的 sensor 都要经过相同的评估测试，调试。通常 camera 的调试会有下面主要几个方面：

清晰度的测试

使用 ISO12233 标板测试



中心垂直分辨率



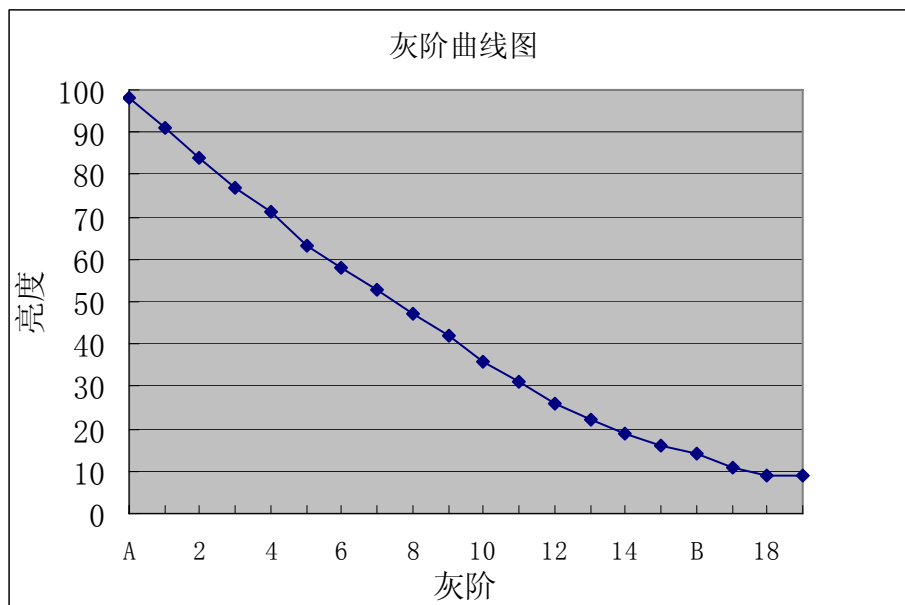
主要看肉眼刚好能够分清线条时候的刻度值。

清晰度主要由 sensor 制作工艺水平和镜头参数决定，但可以通过调试锐度（sharp）来增强清晰度。

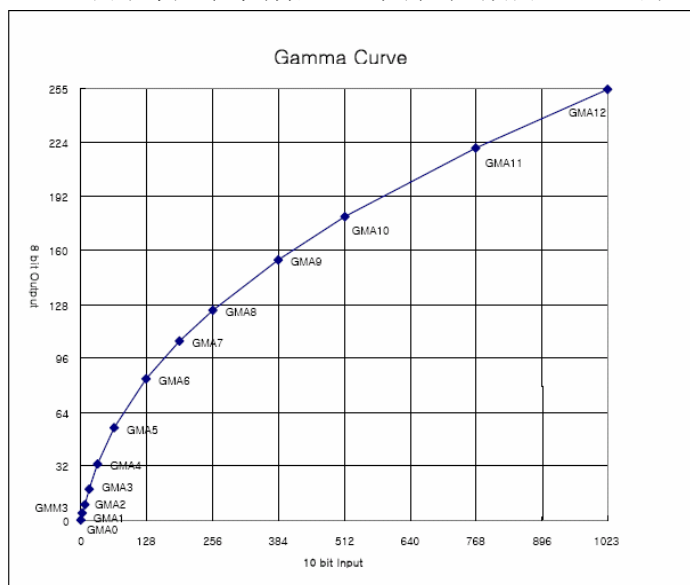
副作用是较高的锐度对图像的平滑性有影响，会使物体边缘特别明显，甚至出现锯齿现象。

灰阶重现测试





一般 sensor 都会有一组寄存器用来调整 gamma 曲线，也就是我们所说的 gamma table，由于本人水平有限，这个调试一般由 micron 的工程师完成。



gamma 曲线图

画面的均匀性以及暗角补偿：



通过上图我们可以发现画面不是很均匀，中心和边缘的亮度有明显的差别，由于镜头的原因，sensor 总是中间的像素暴光比较充分。**Micron** 能够调试 lens shading 来解决这个问题，能将画面调试得更加均匀。

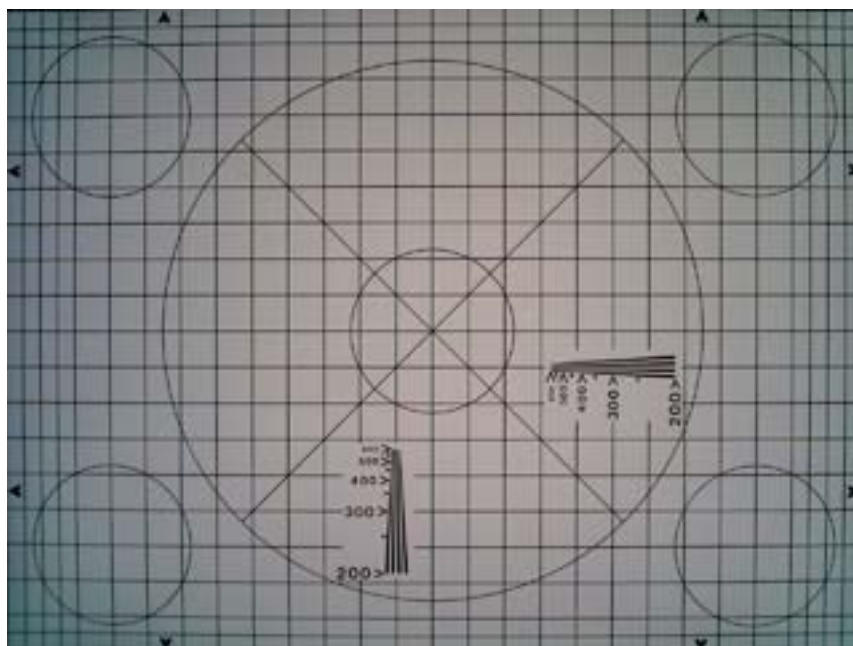
调试过后的图片



畸变:

畸变是由镜头形成的,由 camera 的制造工艺决定,所以无法通过改变 ISP 的设置改善。

下图为测试畸变的样板。



白平衡:

要说到白平衡就要先建立色温的概念

其实在摄影领域,光源大多是根据它们的色温来定义。色温的单位是开尔文,在不同温度下呈现出的色彩就是色温。当一个黑色物体受热后便开始发光,它会先变成暗红色,随着温度的继续升高会变成黄色,然后变成白色,最后就会变成蓝色(大家可以观察一下灯泡中的灯丝,不过由于受到温度的限制,大家一般不会看到它变成蓝色)。总之,这种现象在日常生活中是非常普遍的。



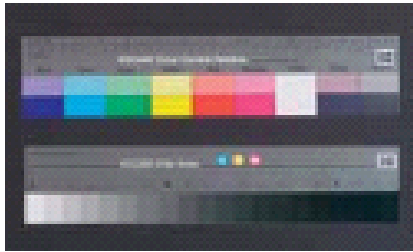
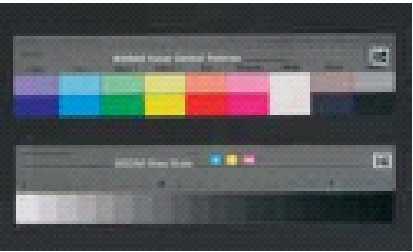
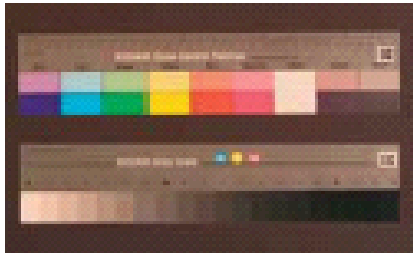
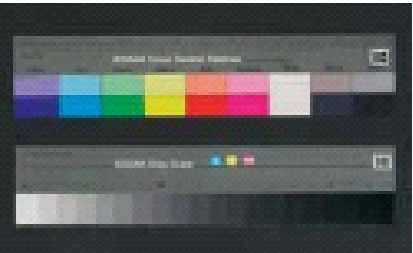
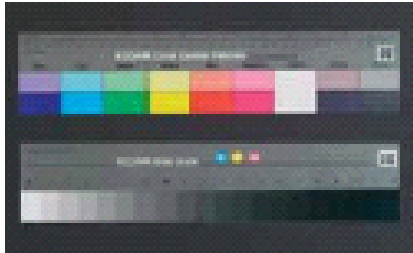
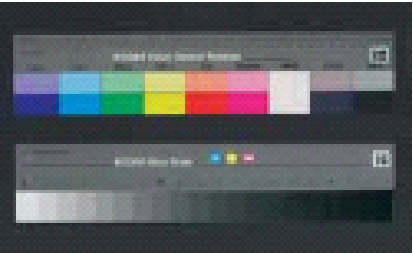
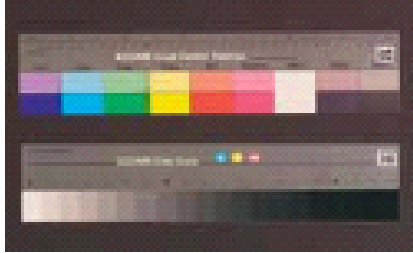
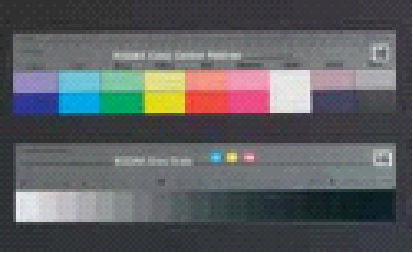
上图就是不同的色温的光源照射下的同一组物体的图片。

人的大脑能仔细分析出从眼睛接受的信号，因而能感知不同的色温(color temperature)来显示相同的白色。但 camera 却不能，在早晨时分的相片偏红，而黄昏时候的却偏黄，就算同一张白纸在不同的环境下被拍摄，如不同的时间，不同的光源，都会出现不同程度的偏差。

调整白平衡，就是要给白色一个定义，能正确记录我们眼睛所看到的颜色。Micron sensor 给出了两种白平衡的控制方法，一种是自动白平衡(AWB)，一种是手动白平衡(MWB)。

自动白平衡为 sensor 内部 ISP 的默认设置，ISP 中有一结构复杂的矩形图，它可决定画面中的白平衡基准点，以此来达到白平衡调校。由于手机 camera 不属于色彩要求很高的照相机范畴，所以我们一般使用自动白平衡，自动白平衡在光源不是特别复杂的时候有较好的效果。

手动白平衡需要自己设置 R, G, B 的 gain 值，micron 给我们的寄存器中有专门设置这三个值的寄存器。

	
光源：白炽灯 白平衡：白炽灯	光源：荧光灯 白平衡：荧光灯
	
光源：白炽灯 白平衡：自动	光源：荧光灯 白平衡：自动
	
光源：白炽灯 白平衡：自定义	光源：荧光灯 白平衡：自定义
	
光源：混合光 白平衡：自动	光源：混合光 白平衡：自定义

作者水平有限，或许会有些不准确之处，欢迎指正批评。