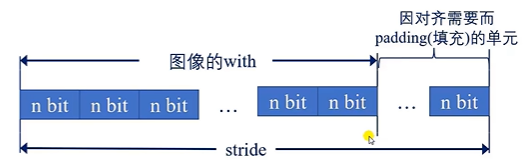


硬件在读取数据时，是按行读取的，在按行读取时，每次就读取stride像素。



图像的实际宽度：with

n表示像素的位宽，可以是8、16、24、38等，主要看是什么格式的图片。不同格式的图片，位宽不一样。

Stride就是步幅，就是读取的宽度（必须是16的整数倍）所以stride一定是大于等于with的。

padding里面的这几个像素就是需要填充的单元。

Stride应该多大呢？

首先需要了解对齐的概念：硬件为了快速访问首地址/跨行访问数据。其内部会进行对齐操作，也就是说内存地址/内存的跨度，必须是 对齐系数 的倍数。

而RV1126的图像处理这块，对齐系数就是16 个像素。

所以要求：跨度，也就是stride，必须要能够被16整除。

Stride和图像的宽度with有什么关系？该怎么设置stride才合理呢?

首先我们知道：图像的宽和图像的大小都是以像素为单位的，图像的操作也是按行操作的。按行读取图像的数据的时候。最理想的状态：每次读取的 就是 图像宽度的大小。（图像的宽度刚好能够被16整除，如1280p）



但如果图像的宽度不能被16整除呢？

当stride < with

如图：若图像是1080，而stride设置为1072.所以在内存缓冲区中，读取的像素就是如右图的样子，即：

第一行的像素被放到了第二行。依此类推，内存缓冲区中，读取的像素就如右下图，排列错乱了。

本来应该处于第一行的像素，处于第二行了...



此时如果没有经过后续的处理，就会产生花纹或条纹：

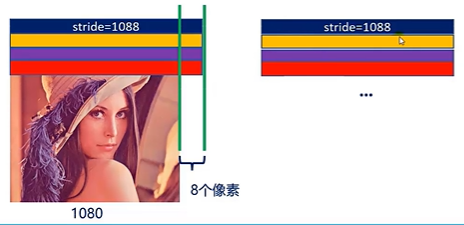


Stride = with

这种是没有产生错乱的，能正常显示的：



Stride > with：



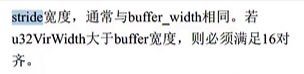
后面会多出8个像素，这8个像素就相当于填充了像素了，在内存缓冲区中，数据就如右图存放的。填充的这8个像素就是无意义的像素。



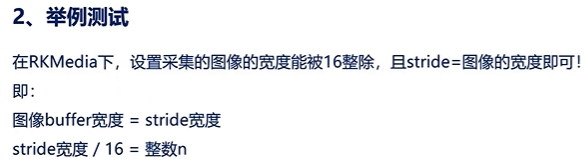
就相当于这8个像素都是0，就会显示为黑色。但对于人眼，8个无意义的像素几乎没有影响，人眼很难分辨出差别。

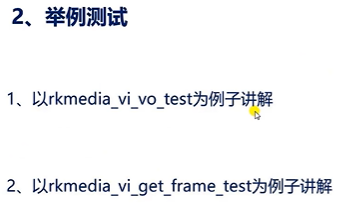
——————————————————————————————————

由官方RKMedia手册得：



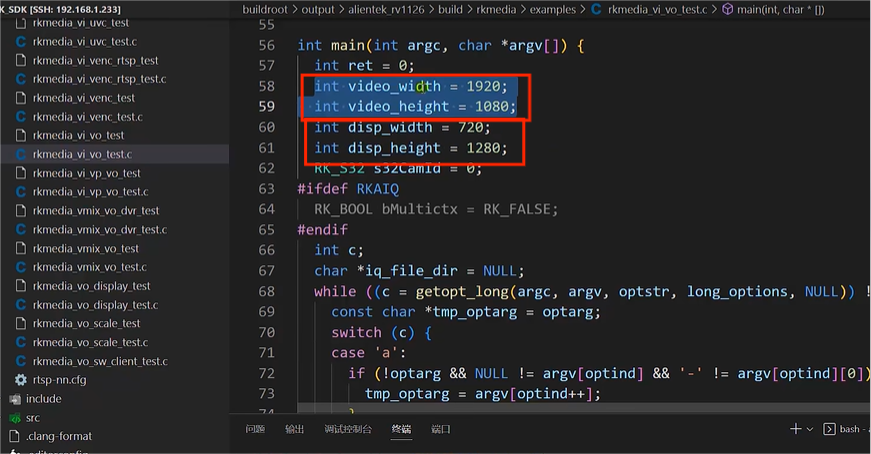
在后面使用RKMedia的时候，省事的做法就是：设置图像的宽高 等于 stride即可。其中stride的宽度要能被16整除：





对应打开：

buildroot/output/alientek\_rv1126/build/rkmeida/examples/rkmedia\_vi\_vo\_test.c



需要注意的就是上面的vido\_width和height。

（下面两个是分辨率，暂时不用管。）

由于vido\_width（图像的宽度）和height可以被16整除，所以采集到的图像是正常显示的。

