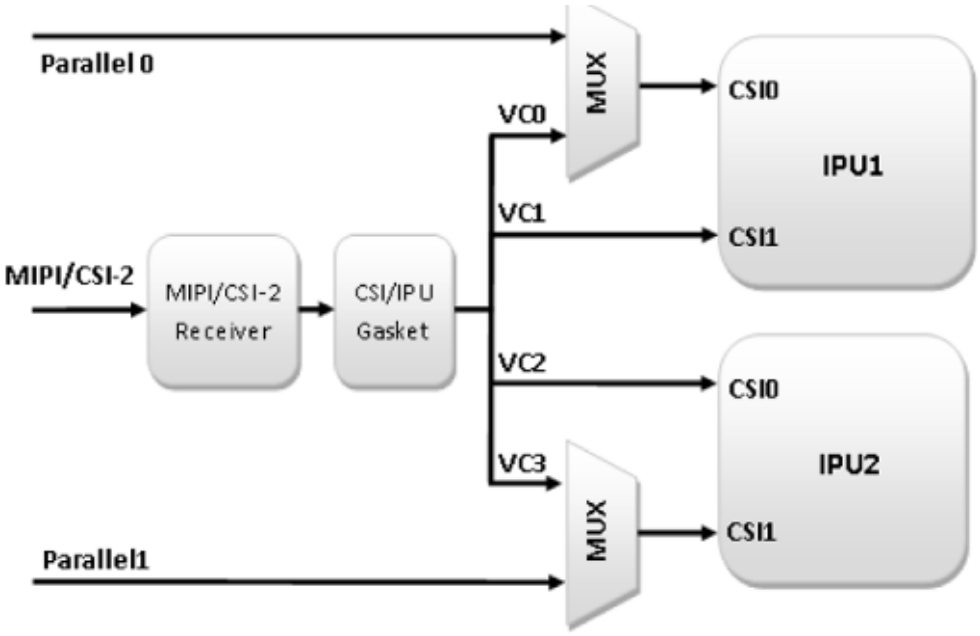
i.MX6Q 捕获原始（bayer）数据和 debayer

原文: https://community.nxp.com/docs/DOC-345148

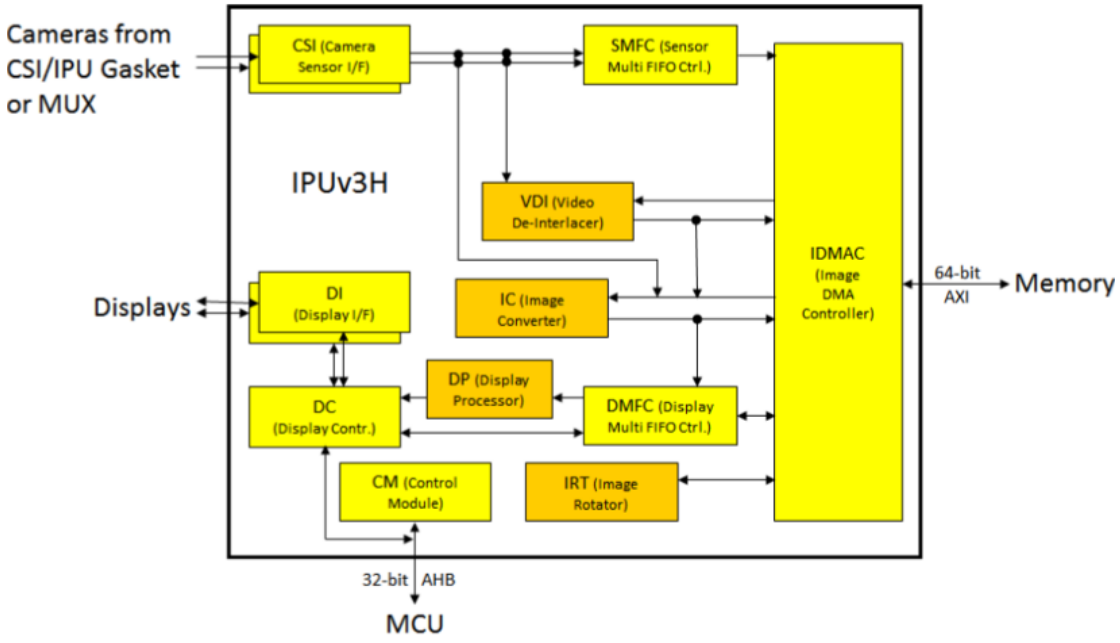
该文档显示：在 i.MX6Q SabreSD 开发板上，配置 ov5640 传感器（并行或 MIPI）以 15fps 的速度输出 5MP（2592x1944）RAW（Bayer）数据，i.MX6Q IPU 捕获 RAW RGB 数据，以及 i.MX6Q GPU debayer RAW 数据然后显示图像。
硬件：i.MX6Q-SabreSD 板，ov5640 传感器。
软件：Linux 4.14.98_2.0.0 BSP，以及此文档中的补丁。

1.在相机传感器侧配置
拜耳滤光片是可以将 RGB 彩色滤光片排列在光电传感器的正方形网格上的彩色滤光片阵列（CFA）。滤镜模式为 50％绿色，25％红色和 25％蓝色，因此也称为 BGGR，RGBG，GRGB 或 RGGB。
ov5640 具有可在 5 兆像素（2592x1944）分辨率下以高达 15 fps 的速度运行的图像阵列。OV5640 支持输出格式: RAW (Bayer), RGB565 / 555/444, CCIR656, YUV422 / 420, YCbCr422 和 compression（压缩）。
为了使 ov5640 以 15fps 的速度输出 500 万像素的 RAW 数据，检查我的补丁 imx6_ov5640_dvp_mipi_raw_capture_driver-4.14.98_2.0.0.diff，此补丁适用于 Linux 的 i.MX BSP 4.14.98_2.0.0 内核代码:
对于并行接口 ov5640，请使用此文件下 driver/media/platform/mxc/capture/ov5640.c 的 ov5640_raw_setting [] 数组。该寄存器设置来自 ov5640 软件应用手册和数据手册。
对于 MIPI 接口 ov5640，请使用此 drivers/media/platform/mxc/capture/ov5640.c 目录下的 ov5640_mipi_raw_setting [] 数组。该寄存器的设置是原始代码的设置（删除 ISP 寄存器设置），MIPI 接口的 PLL 寄存器设置以及某些数据格式寄存器设置的组合。

2.在 i.MX6Q 端配置
i.MX6Q IPU 摄像机端口（CSI-2 模块）支持的数据格式包括 Raw（Bayer），RGB，YUV 4：4：4，YUV 4：2：2 和灰度，每个值最多 16 位。
以下是 i.MX6Q 的相机数据路由:



以下是 i.MX6Q IPU 方框图:



IPU 的 CSI-2，负责同步和打包视频（或通用数据）并将其发送到其他模块。CSI-2 接收到的视频数据可以发送到其他三个模块：SMFC，VDI，IC。

对于 RAW（Bayer）数据捕获，应经过这样的路径：CSI-2-> SMFC-> IDMAC-> DDR 内存这意味着 RAW 数据将作为常规数据接收，请参阅我的补丁中的 IPU_PIX_FMT_GENERIC，而 IPU 无法处理此类数据，它只是接收到 DDR 内存中。

对于 MIPI 接口相机，需要注意的是 i.MX6 侧的 MIPI D-PHY 时钟必须校准为相机传感器 D-PHY 时钟的实际时钟范围，并且校准值必须等于或大于相机传感器时钟，详细信息参见 AN5305。

以 MIPI ov5640 为例:
像素时钟= 2592x1944x15fpsx（1/2 周期/像素）x1.35 空白间隔= 51MHZ
MIPI 数据速率= 51MHZ x 16 位= 816Mb / s
因此 816/2/2 * 2 为 408MHZ 是 i.MX6 侧 D-PHY 时钟。
这里由于一个拜耳像素是 8 位，而 i.MX6 MIPI 数据总线是 16 位，因此上面使用 1/2 个周期/像素。
同时检查 ov5640_mipi_raw_setting [], 你将获得传感器端 D-PHY 时钟约为 672/2 = 336MHZ。
同时检查 AN5305，将 i.MX6 的 MIPI_CSI2_PHY_TST_CTRL1 寄存器设置为 0xC，但在此我仍将其保留为默认 BSP 值 0x14。

3.捕获测试代码
我更改了单元测试 mxc_v4l2_capture.c 以捕获 RAW 数据并将其保存到文件中。检查我的补丁程序 imx6_ov5640_raw_captupre_test_4.14.98_2.0.0_ga.diff，该补丁程序适用于 i.MX Linux 4.14.98_2.0.0 BSP 单元测试代码。
注意用法是:
./cap.out -c 1 -i 1 -fr 15 -m 6 -iw 2592 -ih 1944 -ow 2592 -oh 1944 -f BA81 -d / dev / video1 savefile.dmp
参数-i 1 表示使用 CSI 到 MEM 模式
/ dev / video1 是 MIPI ov5640，/ dev / video0 是并行 ov5640

4.显示 RAW 数据
RAW 数据无法直接显示，需要进行 debayer 处理才能获得每个像素完整的红色，绿色，蓝色。
如果 debayer 进程在 CPU 上运行，则将花费大量 CPU 时间。
为了节省 CPU 时间，可以通过 GPU 完成 debayer。
方法是，将捕获的 RAW 数据作为纹理上传到 GPU，然后 GPU 进行 debayer 处理，然后获得每个像素的全彩，然后显示出来。
要将零内存副本的 RAW 相机数据上传到 GPU，我将使用 i.MX6Q GPU 扩展 GL_VIV_direct_texture。它创建具有直接访问支持的副本。API glTexDirectVIVMap，它支持将用户空间内存或物理地址映射到副本表面。
API glTexDirectVIVMap 需要数据缓冲区的逻辑和物理地址，因此我将从/ dev / mxc_ipu 分配数据缓冲区，它是 dma-buffer 还要获取缓冲区的逻辑/物理地址，然后将它作为 USERPTR 排队到 ipu v4l2 捕获驱动程序，之后出队获得了 RAW 相机数据，然后将其传递给 GPU 进行 debayer。
在 GPU 方面，我将使用“在 GPU 上进行高效，高质量的 Bayer Demosaic 过滤”中的 OpenGL 着色器代码。
检查我的补丁 imx6-5640-debayer-testcode-gpusdk-5.2.0.diff, 该补丁适用于 i.MX GPU SDK 5.2.0 代码。
注意，这里我只做 debayer，没有额外的过程。

5. 已知问题
一件事是 ov5640 以 15fps 的速度输出 5MP，与以 5fps 的速度输出 5MP 相比，在 15fps 的情况下相机数据会有更多的噪点。我调试发现这种噪声似乎来自 ov5640 本身。

参考：

a> https://www.nxp.com/webapp/Download?colCode=IMX6DQRM

b> https://www.nxp.com/webapp/Download?colCode=L4.14.98_2.0.0_MX6QDLSOLOX&appType=license

c> https://github.com/NXPmicro/gtec-demo-framework

d> https://www.nxp.com/docs/zh-CN/application-note/AN5305.pdf

e> ov5640 数据表

f> ov5640 软件应用笔记

g>在 GPU 上进行高效，高质量的 Bayer Demosaic 过滤

https://www.semanticscholar.org/paper/Efficient%2C-High-Quality-Bayer-Demosaic-Filtering-on-McGuire/088a2f47b7ab99c78d41623bdfaf4acdb02358fb

附件：

imx6_ov5640_dvp_mipi_raw_capture_driver-4.14.98_2.0.0.diff.zip

imx6_ov5640_raw_captupre_test_4.14.98_2.0.0_ga.diff.zip

imx6-5640-debayer-testcode-gpusdk-5.2.0.diff.zip