STM32MP1 序列产品连接 MIPI® CSI-2 摄像头

引言

本应用笔记提供关于如何连接 STM32MP1 系列产品与 MIPI CSI-2 摄像头的信息。STM32MP1 系列产品（如 STM32 高性能 MCU）可通过其 DCMI（数码相机模块接口）并行接口寻址 CMOS 摄像头传感器。但是，得益于 STMIPID02 MIPI CSI-2 解串行器离散元件，可以扩展可寻址摄像头传感器的范围，如 MIPI® CSI-2 摄像头（摄像头串行接口）。

多年来，MIPI CSI-2 接口协议早已成为嵌入式传感器领域的标准技术，它主要由移动市场驱动，并被广泛应用于产业市场。MIPI CSI-2 在产业市场具备决定性优势，相比于传统并行接口或 MIPI CPI，减少了引脚数量和成本。

STMIPID02 MIPI CSI-2 解串行器可寻址移动设备和汽车应用中的各种 MIPI CSI-2 摄像头传感器。该功能直接接口免去了与帧解码相关的软件开销要求（就通过 USB 或以太网等方式连接的摄像头而言）。

本应用笔记旨在使用 DH96 Avenger 板演示 STM32MP1 系列产品通过 STMIPID02 MIPI CSI-2 解串行器寻址 5 Mpixel OV5640 MIPI CSI-2 摄像头传感器的能力。两种驱动都可用，并且包含在 STMicroelectronics OpenSTLinux 发行软件包中。就本应用笔记而言，根据 STMIPID02 解串行器规格，只重点考虑使用 D-PHY 接口的 MIPI CSI-2.1 协议。

1 概述

本文档适用于 STM32MP1 系列基于 Arm® Cortex® 内核的微处理器。

提示 *Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。为了更好地理解本文档，需要用到下表中列出的缩略语。*

arm

表 1. 缩略语列表

缩略语	说明
CSI	摄像头串行接口
CPI	摄像头并行接口
MIPI	移动产业处理器接口
DCMI	数字摄像头接口
PMIC	电源管理集成电路
LDO	低压降调节器

2 参考文档

下面的资源是公开的，可以从意法半导体的网站或第三方网站上获得。

表 2. 参考文档

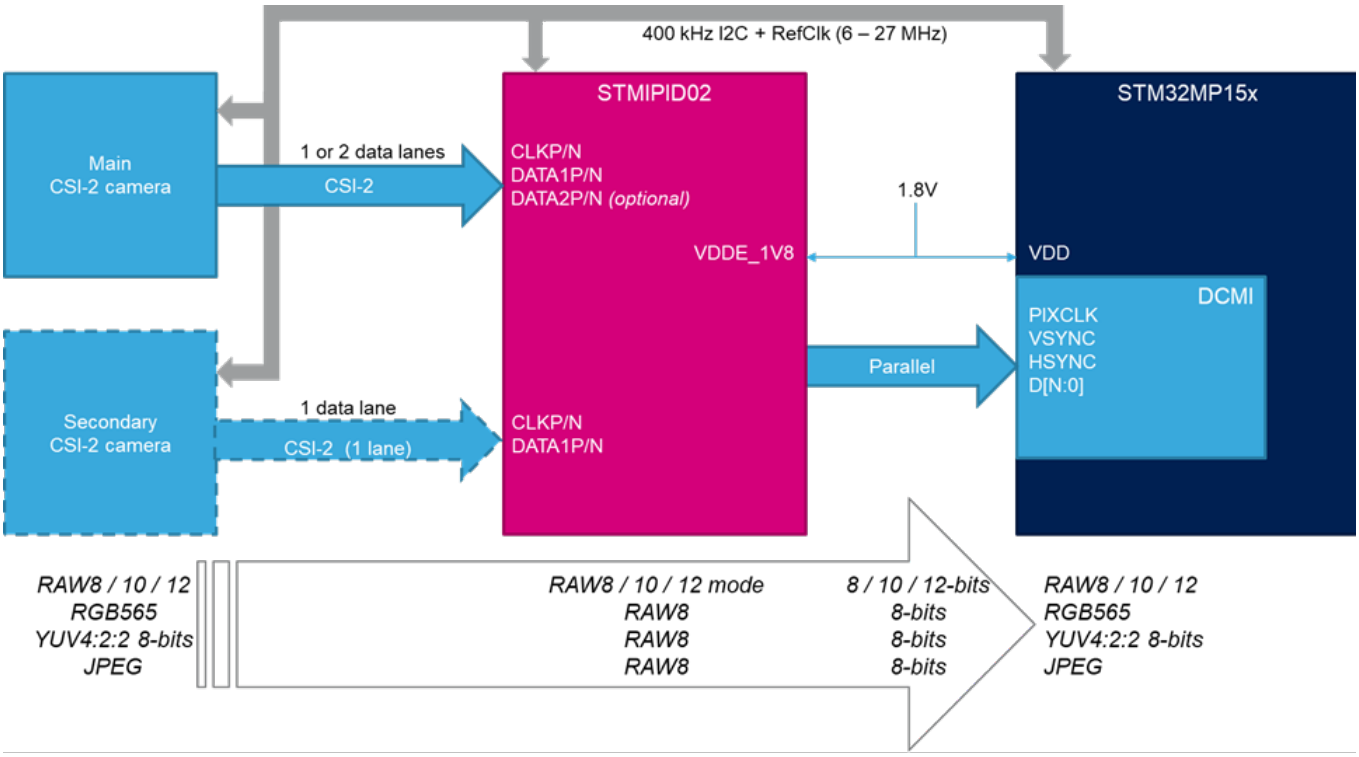
参考编号	文件标题
意法半导体文档[1]	
[R1]	STM32MP15x 数据手册 （DS12500、DS12501、DS12502、DS12503、DS12504、DS12505）
[R2]	STM32MP151x/3x/7x 器件勘误表（ES0438）
[R3]	双模式 MIPI CSI-2 / SMIA CCP2 解串行器（DS12803）
[R4]	STM32MP151、STM32MP153、以及 STM32MP157 系列硬件开发入门（AN5031）
[R5]	基于 Arm® Cortex® 的 STM32 MPU 用户指南： www.wiki.st.com/stm32mpu
[R6]	Linux® V4L2 摄像头框架： www.wiki.st.com/stm32mpu/wiki/V4L2_camera_overview
[R7]	STCubeProgrammer（闪存编程工具）： www.wiki.st.com/stm32mpu/wiki/STM32CubeProgrammer
[R8]	设备树配置： www.wiki.st.com/stm32mpu/wiki/DCMI_device_tree_configuration
[R9]	OpenST Linux distribution：www.wiki.st.com/stm32mpu/wiki/OpenSTLinux_distribution#Reference_source_code
[R10]	OpenSTLinux 发行软件包的目录结构： www.wiki.st.com/stm32mpu/wiki/Example_of_directory_structure_for_Packages
[R11]	STM32CubeProgrammer 软件工具：www.st.com/en/development-tools/stm32cubeprog.html
[R12]	www.wiki.st.com/stm32mpu/wiki/How_to_populate_the_SD_card_with_dd_command
[R13]	wiki.st.com/stm32mpu/wiki/How_to_use_USB_mass_storage_in_U-Boot
[R14]	wiki.st.com/stm32mpu/wiki/Yavta
[R15]	wiki.st.com/stm32mpu/wiki/I2C_i2c-tools
[R16]	wiki.st.com/stm32mpu/wiki/GStreamer_overview
开源软件资源[2]	
[R17]	www.arrow.com (d3cameramezzov5640/d3-engineering)
[R18]	DH96 板信息：www.dh-electronics.com (/index.php/Avenger96)
[R19]	www.github.com
[R20]	www.github.com (dh-electronics/manifest-av96)
[R21]	www.dh-electronics.com (/index.php/Avenger96_Image_Programming)

1. *见 www.st.com。如需详细信息，请与意法半导体联系。*
2. *此 URL 属于第三方。它在文档发布时处于活动状态，但意法半导体对 URL 或参考材料的任何变更、转移或停用不承担责任。*

3 STM32MP1 系列产品与 STMIPID02 MIPI CSI-2 解串器的接口连接

STM32MP1 系列 MPU 系列产品不在本机实现 MIPI CSI-2 接口，而是内置基于 MIPI CPI 接口的 DCMI 并行端口。可通过 STMIPID02 MIPI CSI-2 解串行器对其进行连接，以便寻址任何兼容的 MIPI CSI-2 摄像头传感器设备。STMIPID02 MIPI CSI-2 解串行器的一端连接到 MIPI CSI-2 摄像头，另一端则连接到 STM32MP1 系列 DCMI 12 位数据并行接口。框图总览如下所示。

图 1. 框图总览



3.1 MIPI CSI-2 与 MIPI CPI 接口的比较

请注意，与 MIPI CPI 接口相比，MIPI CSI-2 节省了引脚布局。MIPI CPI 数据端口需要至少 8 条数据线（最多 12 条数据线）、1 个时钟和 2 条同步线，而 MIPI CSI-2 数据端口的每个通道需要 2 线差分对，还需要时钟通道。

3.2 电源的注意事项

考虑到 STMIPID02 解串行器桥外部供电引脚的电压被限制在 1.8 V，为避免对 DCMI 接口时钟和 I^2C 信号使用电平转换器，STM32MP1 系列产品的供电电压必须是 V_{DD} = 1.8 V（而不是标称的 3.3 V）。STM32MP1 系列产品的所有不同电压均通过外部 PMIC（电源管理集成电路）模块供应。

整体示意图详见 DH96 板信息 [R18]。为了给 STM32MP1 系列产品配置 V_{DD} = 1.8 V 的供电电压，请参见 [R4]。

对于 OV5640 摄像头传感器，I/O 供电电压 V_{DD} 和 LDO（低压降调节器）外部电源电压均设置为 1.8 V。对于模拟逻辑，还必须提供 2.8 V 电压和外部电源。

3.3 STM32MP1 系列产品通过 DCMI 实现的视频吞吐率性能

采用 D-PHY 时，MIPI CSI-2.1 接口理论上可以达到每通道最高 2.5 Gbyte/s 的数据吞吐率。在并行接口上很难达到这一水平，原因首先是通用器件（如 STM32MP1 系列产品，只有 MIPI CPI 接口）上的 I/O 引脚压摆率限制。其二，MPU 需要足够快地处理大量数据，以便维持摄像头的连续帧率。

例如，5 Mpixel 传感器的每像素位数为 16 位，帧率为 30 帧/s，得出连续处理的数据吞吐率为 300 Mbyte/s。在并行接口上很难实现这个目标。因此，必须降低传感器图像数据吞吐率，方法是调整图像帧率、分辨率和像素深度（或结合使用）。

从 OV5640 传感器到 STM32MP1 系列 MPU，再到 STMIPID02 解串行器桥，可以连续用以下分辨率和帧率采集图像。

- 720 p 1280 × 720 RGB 565 27 fps
- 720 p 1280 × 720 YUYV 27 fps
- 720 p 1280 × 720 JPEG 27 fps
- HD 1920 × 1080 RGB 565 13 fps
- HD 1920 ×1080 YUYV 13 fps
- HD 1920 ×1080 JPEG 6 fps
- 5 Mpixel 2592 × 1944 RGB565 3 fps
- 5 Mpixel 2592 × 1944 YUYV 3 fps
- 5 Mpixel 2592 ×1944 JPEG 3 fps

达到的最高性能为 24 Mpixel/s，相当于帧率为 18.5 fps 的 1.3 Mpixel。如前文所述，这一限制主要源于 DCMI 内部延迟限制的影响。

3.4 STMIPID02 Linux 驱动

STMIPID02 MIPI CSI-2 解串行器桥用于寻址各种面向消费品市场特别是移动电话应用的 MIPI CSI-2 传感器。为了满足人工智能领域日益增长的将这类传感器从产业向 IoT（物联网）市场转化的需求，STMIPID02 驱动已向上同步到 Linux 社区。在基于 Linux 的应用中可以免费获取它。STMIPID02 桥驱动包含在 STMicroelectronics OpenSTLinux 交付封装中（1.1.0 及以上版本）。

4 综合应用

摄像头演示程序基于 OpenSTLinux 发行软件包，是 GTK 演示启动器应用的一部分。它被移植到 DH96 板上，DH96 Avenger 板配备 STM32MP157A 和 STMIPID02，而 D3 Engineering DesignCore® 摄像头中间板则配有摄像头传感器 OV5640。

4.1 DH Avenger96 板概述

DH96 Avenger 母板集成了：

- 包含 STM32MP157AAC 微处理器的 ADH Core SOM 模块
- STPMIC1A 电源模块
- 2 MB × 512 MB 的 DDR3L RAM
- STMIPID02 解串行器桥
- 2-MB SPI 启动闪存
- 连接和扩展连接器，用于连接 D3 DesignCore 摄像头中间板 OV5640。

有关详细信息，请参见[R18]。

4.2 DH D3 Engineering DesignCore 板概述

本款中间板旨在让 Avenger96 板通过高速和低速扩展连接器适配 STM32MP157 系列。 由此便可通过 MIPI CSI-2 连接 OV5640 模块照相机，从而达到评估目的。

此板件可连接串行控制台，例如 GPIO PD1 和 Pb2（对应于 UART4_TX 和 UART4_RX），并用于显示 Linux 内核和启动阶段。选配 USB/UART 桥可通过这些引脚连接控制台和主机 PC。

4.3 构建板映像

OpenSTLinux 发行软件包面向意法半导体应用板（STM32MP157C-DK2 和 STM32MP1 系列 EVAL 板），提供了练习使用 STM32MP157 系列嵌入式外设所需的驱动、库、工具和示例。

用于第三方 DH96 Avenger 和 D3 Engineering 摄像头板的软件包基于 OpenSTLinux 发行软件包。但是，需要针对 STM32MP157 系列 DCMI 打补丁，然后才能通过 STMIPID02 MIPI CSI-2 解串行器桥本地寻址 MIPI CSI-2 摄像头传感器。

下面列出了多个用于构建板映像的选项，具体取决于是否已经下载了 OpenSTLinux。

关于映像编程，可以从相关的维基百科页面找到 ST 网站 *www.st.com* 提供的许多选项。本应用笔记中详细介绍了主要选项。

4.3.1 从 DH96 GitHub 存储库获取 manifest-av96

如果主机上没有安装 OpenSTLinux 发行软件包，那么下载最新的 Git 存储库 manifest.xml 文件（其中描述了目录结构和源文件链接）可能是最直接的做法。GitHub 网站上提供了可供下载的已打补丁的分发软件，请参见 [R20]。

除了 OpenSTLinux 发行软件包资源，Yocto manifest 还包含 DH Avenger96 板元层“meta-av96”。一个完全构建的源环境可即时安装到主机上，通过 STMIPID02 解串行器桥生成面向 DH96 Avenger 板和 OV5640 摄像头传感器的建成映像。

请参见 [R10] 获取 OpenSTLinux 发行软件包目录结构的构建指南。按照 [R20] 中的描述，应用下列指令构建板映像。

```
PC $> cd <Distribution Package directory>/<distribution version>/
PC $> repo init -u https://github.com/dh-electronics/manifest-av96 -b thud
PC $> repo sync
PC $> source layers/meta-arrow/scripts/init-build-env.sh
PC $> bitbake av96-weston
```

4.3.2 在 OpenSTLinux 发行软件包上添加 meta-av96 层

如果之前已在主机上安装了 OpenSTLinux 发行软件包，则另一个选择是在 OpenSTLinux 分发套件的上方添加 Yocto 板元层 « meta-av96 »。

要实现这一点，在构建新的映像前，必须首先在本地克隆 DH Avenger96 板元层。

4.3.2.1 克隆 Git AV96 存储库层

```
PC $> cd <Distribution Package directory>/<OpenSTLinux distribution>/layers
PC $> git clone https://github.com/dh-electronics/meta-av96 -b thud
```

4.3.2.2 *为新机器设置 Yocto Bitbake 环境*

```
PC $> cd ../
PC $> META_LAYER_ROOT=layers DISTRO=openstlinux-weston MACHINE=stm32mp1-av96
PC $> source layers/meta-st/scripts/envsetup.sh
```

4.3.2.3 *添加板元层和运行 Bitbake*

```
PC $> bitbake-layers add-layer ../layers/meta-av96
PC $> bitbake stm32mp1-av96
```

创建板映像可能需要几个小时和约 20 GB 磁盘空间。创建的映像目录如下：

`<Distribution-Package build directory>/tmp-glibc/deploy/images/stm32mp1-av96`

4.3.3 STM32CubeProgrammer 软件工具

使用这款一体化软件工具，可以轻松地将二进制映像迁移到任何闪存器件上。此工具可以从 [R11] 下载。

为了更快地完成编程，建议将板硬件启动开关选择为 DFU（或 USB 启动模式），如下表所示。

表 3. STM32CubeProgrammer 软件

自举模式	注释	启动 2（开关 3）	启动 1（开关 2）	启动 0（开关 1）
UART 和 USB	USB OTG	1	1	0

在完成二进制映像的编程后：

1. 将硬件启动开关选择为 SDCard 启动模式。
   a. Boot2 = 1
   b. Boot1 = 0
   c. Boot0 = 1

4.3.4 其他编程工具

有其他编程工具可供选择。

- 如需使用传统的“dd”指令将二进制映像编程到 SDcard 上，请参见 [R12] 中的指南。
- 但是，如果必须在 NOR-Flash（TF-A、U-boot）和 e-mmc（Linux）之间对二进制映像分区进行分配，建议将整个二进制图像编程到 SDcard 中。更多详细信息，请参见[R20]。
- 如需使用 U-boot 打开 USB 大容量存储设备，请参见 [R13]。

如果有 U-boot 可用，从而 SDCard 或板的非易失性存储器上已有两个原始闪存板分区，则适用此方法。

4.4 启动板映像和摄像头预览屏幕

在启动引脚更改为 SDcard（或最终用户默认的映像设置）后：

1. 连接 HDMI® 以显示显示器或 TV。
2. 弹出意法半导体演示程序。
3. 插入 USB 鼠标。
4. 选择摄像头图标，流传输来自 OV5640 MIPI CSI-2 摄像头模块的实时视频。

下图显示了用黄色高亮显示摄像头图标的界面。

图 2. 摄像头图标

4.4.1 V4L2 指令

有一组内置的用来与摄像头传感器通信的 V4L2 指令。其中最常用的指令是 `v4l2-ctrl`。它可以检索所有摄像头传感器属性，检查 V4L2 设备节点设置，以及向控制台发送指令。指令的具体信息如下。

列出摄像头控制菜单

```
root@stm32mp1-av96:~# v4l2-ctl -L

User Controls

                     contrast 0x00980901 (int)    : min=0 max=255 step=1 default=0 value=0
flags=slider
                   saturation 0x00980902 (int)    : min=0 max=255 step=1 default=64
value=64 flags=slider
                          hue 0x00980903 (int)    : min=0 max=359 step=1 default=0 value=0
flags=slider
        white_balance_automatic 0x0098090c (bool)   : default=1 value=1 flags=update
                  red_balance 0x0098090e (int)    : min=0 max=4095 step=1 default=0
value=0 flags=inactive, slider
                 blue_balance 0x0098090f (int)    : min=0 max=4095 step=1 default=0
value=0 flags=inactive, slider
                     exposure 0x00980911 (int)    : min=0 max=65535 step=1 default=0
value=972 flags=inactive, volatile
                gain_automatic 0x00980912 (bool)   : default=1 value=1 flags=update
                          gain 0x00980913 (int)    : min=0 max=1023 step=1 default=0
value=19 flags=inactive, volatile
               horizontal_flip 0x00980914 (bool)   : default=0 value=0
                 vertical_flip 0x00980915 (bool)   : default=0 value=0
           power_line_frequency 0x00980918 (menu)   : min=0 max=3 default=1 value=1
                                0: Disabled
                                1: 50 Hz
                                2: 60 Hz
                                3: Auto

Camera Controls

                  auto_exposure 0x009a0901 (menu)   : min=0 max=1 default=0 value=0
flags=update
                                0: Auto Mode
                                1: Manual Mode

Image Processing Controls

                 link_frequency 0x009f0901 (intmenu): min=0 max=0 default=0 value=0
flags=read-only
                                0: 384000000 (0x16e36000)
                   test_pattern 0x009f0903 (menu)   : min=0 max=4 default=0 value=0
                                0: Disabled
                                1: Color bars
                                2: Color bars w/ rolling bar
                                3: Color squares
                                4: Color squares w/ rolling bar
```

显示 DCMI 驱动信息

```
root@stm32mp1-av96:~# v4l2-ctl -d /dev/video0 -D
Driver Info:
        Driver name      : stm32-dcmi
        Card type        : STM32 Camera Memory Interface
        Bus info         : platform:dcmi
        Driver version   : X.Y.Z
        Capabilities     : 0x85200001
                Video Capture
                Read/Write
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps      : 0x05200001
                Video Capture
                Read/Write
                Streaming
                Extended Pix Format
Media Driver Info:
        Driver name      : stm32-dcmi
        Model            : stm32-dcmi
        Serial           :
        Bus info         : platform:stm32-dcmi
        Media version    : X.Y.Z
        Hardware revision: 0x00000000 (0)
        Driver version   : X.Y.Z
Interface Info:
        ID               : 0x03000003
        Type             : V4L Video
Entity Info:
        ID               : 0x00000001 (1)
        Name             : stm32_dcmi
        Function         : V4L2 I/O
        Flags         : default
        Pad 0x01000002   : 0: Sink
          Link 0x02000009: from remote pad 0x1000008 of entity 'st-mipid02 1-0014': Data,
Enabled, Immutable
```

流传输用 RGB 格式采集的图像并保存在文件中

```
root@stm32mp1-av96:~# v4l2-ctl --set-fmt-video=width=1280,height=720,pixelformat=RGBP --
stream-mmap --stream-count=1 --stream-to=file.raw
<
root@stm32mp1-av96:~# ls -ltr file.raw
-rw-r--r-- 1 root root 1843200 Jun 17 17:18 file.raw
```

执行全屏预览

```
root@stm32mp1-av96:~# gst-launch-1.0 v4l2src ! "video/x-raw, width=1280, Height=720,
framerate=(fraction)15/1" ! queue ! autovideosink -e
```

关于 V4L2 Linux 框架和摄像头驱动的综合概述，请参见 [R6]。

4.4.2 应用指令

应用指令控制摄像头的分辨率、帧率和格式，以便基于 V4L2 框架、GStreamer 或 Yavta 通过 HDMI® 或 LCD 屏幕显示图像流。它们由 Linux 提供，包含在 OpenSTLinux 发行软件包中。

4.4.2.1 GStreamer

GStreamer 是一种开源多媒体框架，用于在各种操作系统（如 GNU/Linux 或 Windows）中处理视频和声音。它基于串接管道指令，可实时处理视频流。

如果板映像上没有安装 Gstreamer，则从联网板上使用 `apt-get` 指令。

```
root@stm32mp1-av96:~# apt-get update
…
Reading package lists... Done

root@stm32mp1-av96:~# apt-get install GStreamer
…
Preparing to unpack .../GStreamer_0.10.36-r2_armhf.deb ...
Unpacking GStreamer (0.10.36-r2) ...
Setting up GStreamer (0.10.36-r2) ...
```

该指令以 30 fps 的帧率预览 640 × 480 图像流，使用 HDMI 线缆将显示器连接到 Avenger96 板。

```
root@stm32mp1-av96:~# gst-launch-1.0 v4l2src device=/dev/video0 ! "video/x-raw,
width=640,height=480,framerate=30/1” ! waylandsink &
```

该指令采集 3 张 640 x 480 的 VGA JPEG 图像并保存到文件中。

```
root@stm32mp1-av96:~# gst-launch-1.0 v4l2src num-buffers=3 ! "image/jpeg, width=640,
height=480" ! queue ! multifilesink location=pic%05d.jpeg
```

该指令检查生成的输出文件格式是否正确。

```
root@stm32mp1-av96:~# gst-typefind-1.0 pic00000.jpeg
pic00000.jpeg - image/jpeg, width=(int)640, height=(int)480, sof-marker=(int)0
```

有关详细信息，请参见[R16]。

4.4.2.2 Yavta

Yavta 测试应用基于 V4L2 框架，用于测试、控制和调试摄像头传感器。指令的具体信息如下。

该指令将列出所有传感器设置、支持的视频格式和帧率。

```
root@stm32mp1-av96:~# yavta -l --enum-formats --enum-inputs /dev/video0
Device /dev/video0 opened.
Device `STM32 Camera Memory Interface' on `platform:dcmi' is a video output (without
mplanes) device.
--- User Controls (class 0x00980001) ---
control 0x00980901 `Contrast' min 0 max 255 step 1 default 0 current 0.
control 0x00980902 `Saturation' min 0 max 255 step 1 default 64 current 64.
control 0x00980903 `Hue' min 0 max 359 step 1 default 0 current 0.
control 0x0098090c `White Balance, Automatic' min 0 max 1 step 1 default 1 current 1.
control 0x0098090e `Red Balance' min 0 max 4095 step 1 default 0 current 0.
control 0x0098090f `Blue Balance' min 0 max 4095 step 1 default 0 current 0.
control 0x00980911 `Exposure' min 0 max 65535 step 1 default 0 current 885.
control 0x00980912 `Gain, Automatic' min 0 max 1 step 1 default 1 current 1.
control 0x00980913 `Gain' min 0 max 1023 step 1 default 0 current 248.
control 0x00980914 `Horizontal Flip' min 0 max 1 step 1 default 0 current 0.
control 0x00980915 `Vertical Flip' min 0 max 1 step 1 default 0 current 0.
control 0x00980918 `Power Line Frequency' min 0 max 3 step 1 default 1 current 1.
  0: Disabled
  1: 50 Hz (*)
  2: 60 Hz
  3: Auto
--- Camera Controls (class 0x009a0001) ---
control 0x009a0901 `Auto Exposure' min 0 max 1 step 1 default 0 current 0.
  0: Auto Mode (*)
  1: Manual Mode
--- Image Processing Controls (class 0x009f0001) ---
control 0x009f0901 `Link Frequency' min 0 max 0 step 1 default 0 current 0.
  0: 384000000 (*)
control 0x009f0903 `Test Pattern' min 0 max 4 step 1 default 0 current 0.
  0: Disabled (*)
  1: Color bars
  2: Color bars w/ rolling bar
  3: Color squares
  4: Color squares w/ rolling bar
15 controls found.
- Available formats:
        Format 0: JPEG (4745504a)
        Type: Video capture (1)
        Name: JFIF JPEG
        Frame size: 176x144 (1/15, 1/30)
        Frame size: 320x240 (1/15, 1/30)
        Frame size: 640x480 (1/15, 1/30, 1/60)
        Frame size: 720x480 (1/15, 1/30)
        Frame size: 720x576 (1/15, 1/30)
        Frame size: 1024x768 (1/15, 1/30)
        Frame size: 1280x720 (1/15, 1/30)
        Frame size: 1920x1080 (1/15, 1/30)
        Frame size: 2592x1944 (1/15, 1/30)

        Format 1: UYVY (59565955)
        Type: Video capture (1)
        Name: UYVY 4:2:2
        Frame size: 176x144 (1/15, 1/30)
        Frame size: 320x240 (1/15, 1/30)
        Frame size: 640x480 (1/15, 1/30, 1/60)
        Frame size: 720x480 (1/15, 1/30)
        Frame size: 720x576 (1/15, 1/30)
        Frame size: 1024x768 (1/15, 1/30)
        Frame size: 1280x720 (1/15, 1/30)
        Frame size: 1920x1080 (1/15, 1/30)
        Frame size: 2592x1944 (1/15, 1/30)

        Format 2: YUYV (56595559)
        Type: Video capture (1)
        Name: YUYV 4:2:2
        Frame size: 176x144 (1/15, 1/30)
        Frame size: 320x240 (1/15, 1/30)
        Frame size: 640x480 (1/15, 1/30, 1/60)
        Frame size: 720x480 (1/15, 1/30)
```

```
        Frame size: 720x576 (1/15, 1/30)
        Frame size: 1024x768 (1/15, 1/30)
        Frame size: 1280x720 (1/15, 1/30)
        Frame size: 1920x1080 (1/15, 1/30)
        Frame size: 2592x1944 (1/15, 1/30)

        Format 3: RGB565 (50424752)
        Type: Video capture (1)
        Name: 16-bit RGB 5-6-5
        Frame size: 176x144 (1/15, 1/30)
        Frame size: 320x240 (1/15, 1/30)
        Frame size: 640x480 (1/15, 1/30, 1/60)
        Frame size: 720x480 (1/15, 1/30)
        Frame size: 720x576 (1/15, 1/30)
        Frame size: 1024x768 (1/15, 1/30)
        Frame size: 1280x720 (1/15, 1/30)
        Frame size: 1920x1080 (1/15, 1/30)
        Frame size: 2592x1944 (1/15, 1/30)

- Available inputs:
        Input 0: Camera.

Video format: JPEG (4745504a) 320x240 (stride 320) field none buffer size 76800
```

该指令将 10 个帧以默认格式分辨率写到磁盘上。

```
root@stm32mp1-av96:~# yavta -F /dev/video0 --capture=10
Device /dev/video0 opened.
Device `STM32 Camera Memory Interface' on `platform:dcmi' is a video output (without
mplanes) device.
Video format: JPEG (4745504a) 320x240 (stride 320) field none buffer size 76800
8 buffers requested.
length: 76800 offset: 0 timestamp type/source: mono/EoF
Buffer 0/0 mapped at address 0xb6e2e000.
length: 76800 offset: 77824 timestamp type/source: mono/EoF
Buffer 1/0 mapped at address 0xb6e1b000.
length: 76800 offset: 155648 timestamp type/source: mono/EoF
Buffer 2/0 mapped at address 0xb6e08000.
length: 76800 offset: 233472 timestamp type/source: mono/EoF
Buffer 3/0 mapped at address 0xb6df5000.
length: 76800 offset: 311296 timestamp type/source: mono/EoF
Buffer 4/0 mapped at address 0xb6de2000.
length: 76800 offset: 389120 timestamp type/source: mono/EoF
Buffer 5/0 mapped at address 0xb6dcf000.
length: 76800 offset: 466944 timestamp type/source: mono/EoF
Buffer 6/0 mapped at address 0xb6dbc000.
length: 76800 offset: 544768 timestamp type/source: mono/EoF
Buffer 7/0 mapped at address 0xb6da9000.
Warning: bytes used 6144 != image size 76800 for plane 0
0 (0) [-] none 0 6144 B 398.623775 398.623910 24.408 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
1 (1) [-] none 1 6144 B 398.657082 398.657199 30.024 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
2 (2) [-] none 2 6144 B 398.690402 398.690512 30.012 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
3 (3) [-] none 3 6144 B 398.723708 398.723812 30.025 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
4 (4) [-] none 4 6144 B 398.757021 398.757134 30.018 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
5 (5) [-] none 5 6144 B 398.790334 398.790439 30.018 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
6 (6) [-] none 6 6144 B 398.823651 398.823753 30.015 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
7 (7) [-] none 7 6144 B 398.856968 398.857067 30.015 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
8 (0) [-] none 8 6144 B 398.890280 398.890381 30.019 fps ts mono/EoF
Warning: bytes used 6144 != image size 76800 for plane 0
9 (1) [-] none 9 6144 B 398.923596 398.923693 30.016 fps ts mono/EoF
Captured 10 frames in 0.340887 seconds (29.335169 fps, 180235.280085 B/s).
8 buffers released.
```

关于 Yavta 的更多信息，请参见 [R14]。

5 与另一种 MIPI CSI-2 摄像头传感器的连接指南

本节提供的指导信息可帮助最终用户构建自己的应用，以及用另一种型号的 MIPI CSI-2 摄像头传感器替换 OV5640 摄像头传感器。

在选择摄像头传感器时，必须注意的是，STMIPID02 和 STM32MP1 系列 DCMI 都不能解码来自摄像头传感器的视频帧。为避免帧解码产生的 CPU 开销，新的摄像头传感器必须像 OV5640 摄像头传感器一样内置图像传感器处理器（ISP），否则就需要将处理分散到应用板的离散元件中。

OpenSTLinux 扩展包 交付中提供了 STMIPID02 桥和 OmniVision® OV5640 摄像头传感器驱动。默认情况下，不会包含新的摄像头传感器的 Linux 驱动。

本节将描述在最终用户应用中用另一个摄像头传感器替换 OV5640 摄像头传感器的必要步骤。必须将下列与 OV5640 摄像头传感器相关的所有步骤转移到新的摄像头传感器上。

5.1 I²C 探测

在将新的摄像头插到板子上后，首先要能够通过 I²C 端口对其进行访问。摄像头传感器驱动就是基于这一点，允许访问传感器设备寄存器。

在该步骤之前，必须通过 GPIO 驱动复位和掉电信号。该步骤由直接执行或由具有正确驱动极性（依据传感器规格）I²C 控制的板 GPIO 扩展器执行。

5.1.1 为摄像头传感器驱动打补丁以用于调试

由于电源和时钟是由摄像头传感器驱动动态控制的，无论是否探查模块，都必须为驱动打临时补丁，以便维持探查驱动模块时的电源链路。在摄像头通电并运行时，可以使用 i2ctools 或调试工具读取或写入寄存器值。OV5640 摄像头传感器的驱动位于以下位置：

`<distri_pack_dir>/build-openstlinuxweston-stm32mp1-av96/tmp-glibc/work-shared/stm32mp1-av96/kernel-source/drivers/media/i2c/ov5640.c`

该补丁在探查 OV5640 摄像头模块时禁用关断功能，在新的摄像头驱动上也可以复制这一点。

```
-    ov5640_power(sensor, false);
-    regulator_bulk_disable(OV5640_NUM_SUPPLIES, sensor->supplies);

 xclk_off:
     clk_disable_unprepare(sensor->xclk);
     return ret;
@@ -1878,9 +1878,9 @@ static int ov5640_set_power_on(struct ov5640_dev *sensor)

 static void ov5640_set_power_off(struct ov5640_dev *sensor)
 {
-    ov5640_power(sensor, false);
-    regulator_bulk_disable(OV5640_NUM_SUPPLIES, sensor->supplies);
-    clk_disable_unprepare(sensor->xclk);
 }

 static int ov5640_set_power(struct ov5640_dev *sensor, bool on)
@@ -2886,8 +2886,8 @@ static int ov5640_probe(struct i2c_client *client,
     mutex_init(&sensor->lock);

     ret = ov5640_check_chip_id(sensor);
-    if (ret)
-        goto entity_cleanup;

     ret = ov5640_init_controls(sensor);
     if (ret)
@@ -2897,6 +2897,7 @@ static int ov5640_probe(struct i2c_client *client,
     if (ret)
         goto free_ctrls;

+    dev_info(dev, " probe OK\n");
     return 0;
```

5.1.2 为 STMIPID02 驱动打补丁以用于调试

驱动位于 OpenSTLinux 发行软件包目录：

`<distri_pack_dir>/build-openstlinuxweston-stm32mp1-av96/tmp-glibc/work-shared/stm32mp1-av96/kernel-source/drivers/media/i2c/st-mipid02.c`

与摄像头传感器一样，在调试时，为了探查 STMIPID02 器件，需要强制启用时钟和电源线并禁用复位线。

```
static int mipid02_probe(struct i2c_client *client,
[...]
power_off:
entity_cleanup:
static int mipid02_remove(struct i2c_client *client)
[...]



static void mipid02_apply_reset(struct mipid02_dev *bridge)
{

#if 0
    gpiod_set_value_cansleep(bridge->reset_gpio, 0);
    usleep_range(5000, 10000);
    gpiod_set_value_cansleep(bridge->reset_gpio, 1);
    usleep_range(5000, 10000);
    gpiod_set_value_cansleep(bridge->reset_gpio, 0);
#else
    gpiod_set_value_cansleep(bridge->reset_gpio, 0);
    usleep_range(5000, 10000);
#endif
}
```

5.1.3 重新编译内核模块用于调试

为了简化模块操作，确保 dcmi、stmipid02 和 ov5640 在位于以下位置的 .config 文件中被声明为树外模块：`<distribution-package>/<build>/tmp-glibc/work-shared/stm32mp1-av96/kernel-build-artifacts/`

```
CONFIG_VIDEO_OV5640=m
CONFIG_VIDEO_STM32_DCMI=m
CONFIG_VIDEO_ST_MIPID02=m
```

在应用补丁后，需要重新编译内核模块并将其重载到以下目标上：

```
PC $> bitbake virtual/kernel -C compile
```

在以下目录中更新内核对象 ov5640.ko 和 st-mipid02.ko 或模块：

`<distri_pack_dir>/build-openstlinuxweston-stm32mp1-av96/tmp-glibc/work /stm32mp1_av96-ostl-linux-gnueabi/linux-stm32mp/<kernel_version>/image /lib/modules/<kernel version>/kernel/drivers/media/i2c`

并上传到板 Sdcard rootfs 分区：

`<rootfs>/lib/modules/<kernel version>/kernel/drivers/media/i2c`

为了重新创建用于调试的新映像，以便包含打了补丁的模块，在上一条指令的上方调用如下 Bitbake 指令：

```
PC $> bitbake av96-weston
```

5.1.4 使用 I²C 工具探查模块

I²C-tools 工具包是 OpenSTLinux 发行软件包中内置的 I2C Linux 指令集，可用于探查摄像头传感器和 STMIPID02。

如需列举板子上连接并列出的所有 I²C 总线，执行以下指令：

```
root@stm32mp1-av96:~# i2cdetect -l
i2c-1   i2c         STM32F7 I2C(0x40013000)              I2C adapter
i2c-2   i2c         STM32F7 I2C(0x5c002000)              I2C adapter
i2c-0   i2c         STM32F7 I2C(0x40012000)              I2C adapter
```

如需列出连接到特定 I²C 总线的 I2C 设备寄存器，如 STM32MP1 系列 I2C2 外设对应的 i2c-1，执行以下指令：

```
root@stm32mp1-av96:~# i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- UU -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

UU 表示是当前使用的寄存器地址。由于驱动程序目前正在控制该资源，因此 `i2cdetect` 指令不能探查此地址。例如，I²C 地址 0x3c（在设备树文件中报告）指向 OV5640 摄像头且该摄像头被检测为忙碌状态，原因是 DCMI Linux 驱动已经对其进行了访问。

在将 ov5640.ko 或打了补丁的映像重载到目标上后，在探查模块时，`I2C` 指令可以访问目标。如需检索 OV5640 传感器的 ID 代码（使用 I2C2 slave@0x3c 从传感器寄存器地址 0x300a 读取），使用指令 `i2cget`。

```
root@stm32mp1-av96:~# i2cset -f -y 1 0x3c 0x30 0x0a
root@stm32mp1-av96:~# i2cget -f -y 1 0x3c
0x56
root@stm32mp1-av96:~# i2cget -f -y 1 0x3c
0x40
```

读取值 0x5640 与公布的 OV5640 传感器 ID 代码匹配。使用 `i2ctransfer` 指令会更直接。

```
root@stm32mp1-av96:~# i2ctransfer -y -f 1 w2@0x3c 0x30 0x0a r2
0x56 0x40
```

现在，可以从 STM32MP1 系列产品访问摄像头传感器，并且可以根据传感器软件规格设置摄像头寄存器。

STMIPID02 由 i2c-2 控制，对应于板设备树 @0x14 中的 I2C4。

```
root@stm32mp1-av96:~# i2ctransfer -y -f 2 w2@0x14 0x00 0x14 r1
0x42
```

有关 i2c-tools 的更多信息，请参见 [R15]。

5.2 STMIPID02 MIPI D-PHY 时钟设置

摄像头传感器将 MIPI CSI-2 信号或串行链路驱动到 STMIPID02。这包括连接到 STMIPID02 D-PHY RX 的 D-PHY TX 时钟或时钟通道。此外，STMIPID02 必须根据传感器比特率调整采样时钟通道频率。对于驱动而言，摄像头像素数据率是已知的。它由 V4L2（Linux 框架的视频）进行计算，其中的像素宽度、高度和帧率均作为参数由 API 指令传递给驱动。stmipid02 驱动读取 V4L2_CID_PIXEL_RATE 和 V4L2_CID_LINK_FREQ，以便计算时钟通道频率。

根据使用的帧格式，当每像素位数确定时（就 RGB 和 RAW 而言），可从期望数据率推导，进而直接从 V4L2_CID_PIXEL_RATE 变量推导出时钟通道。

- 其中，MIPI 时钟频率 = 像素时钟频率 × 每像素位数 ÷ ((通道数) ÷ 2)。
- 当每像素位数对驱动而言未知时，如 YUV 或 JPEG 格式，每一帧后面的图像大小可能会有变化。每一行发送后的消隐周期也会变化。

可以使用示波器，在摄像头激活时用高速差分探头测量 MIPI D-PHY 时钟通道。应使用以下公式设置 MIPI CSI-2 数据率：

- MIPI CSI-2 数据率 = (MIPI 时钟频率 × 2) × 数据通道数 ≥ 像素时钟 × 每像素位数。
- 其中，MIPI 时钟频率 = 像素时钟 × 每像素位数 ÷ ((通道数) ÷ 2)。

在 OV5640 摄像头上，可使用示波器测量时钟通道频率：

- 对于 640 × 480 RGB565 15 fps：62 MHz
- 对于 640 × 480 RGB565 30 fps：120 MHz

摄像头输出连续视频流的最大吞吐率为：

- 最大值 24 Mpixel/s，即 48 Mbyte/s 或 384 Mbit/s 两行，或 192 Mbit/s 每行。

对于 STMIPID02 时钟通道 1 寄存器（地址 0x02），下面的设置对应于 OV5640 时钟通道采样。

```
i2ctransfer -y -f 1 w3@0x14 0x00 0x02 0x19
i2ctransfer -y -f 1 w3@0x14 0x00 0x02 0x21
i2ctransfer -y -f 1 w3@0x14 0x00 0x02 0x29
```

如果安装其他摄像头传感器，可能必须重新调整这些设置。

5.3 检查数据信号极性

如果在摄像头传感器和 STM32MP1 系列产品之间设置了 I²C 链路，则 STM32MP1 系列产品、STMIPID02 和摄像头传感器之间的所有时钟和数据信号极性都必须匹配。这些信号可通过硬件板测试点进行物理探查，并使用 I²C 寄存器写指令进行微调。对于通过 MIPI CSI-2 总线传播的摄像头传感器数据和时钟信号，在 STMIPID02 解串行器桥之后进行探查。当探查的所有信号的极性相同时，可以在板设备树中报告信号极性设置，在运行时间可以使用下列指令进行更改。

脚本说明

1. 使用下列必备脚本。

```
echo "#!/bin/bash" > dtdumpentry.sh;echo "hexdump -e '\"=\"'  -e '20/1 \"%c\"\"\t\"' -e
'20/1 \"%02x\"\"\n\"' \$1" >> dtdumpentry.sh;chmod +x dtdumpentry.sh
 echo "#!/bin/bash" > dtdump.sh;echo "find  \$1* -type f -print0 -exec ./dtdumpentry.sh
{} \;" >> dtdump.sh;chmod +x dtdump.sh
```

2. 创建下列可执行脚本。

```
rm devicetree.txt
 echo "[devicetree]" >> devicetree.txt
 echo "|-[dcmi]" >> devicetree.txt
 ./dtdump.sh /proc/device-tree/soc/dcmi | sed 's/\/proc\/device-tree\/soc\//| |-/' >>
devicetree.txt
 echo "|" >> devicetree.txt
 echo "|-[camera:" | tr -d "\n" >> devicetree.txt
 cat /proc/device-tree/soc/i2c*/camera*/compatible >> devicetree.txt
 echo "]" >> devicetree.txt
 ./dtdump.sh /proc/device-tree/soc/i2c*/camera* -type f -print0 -exec ./dtdump.sh {} \;
| sed 's/\/proc\/device-tree\/soc\//| |-/' >> devicetree.txt
 echo "" >> devicetree.txt
 cat devicetree.txt
```

3. 运行创建的脚本。

   新创建的脚本列出了摄像头设备树，以便从 Linux 的角度查看 DCMI 和摄像头传感器端口是如何配置的。它还检查信号极性是否配置正确。

脚本结果

DCMI 端口信号说明。

```
root@stm32mp1-av96:~# ./runme.sh
[devicetree]
|-[dcmi]
…
| |-dcmi@4c006000/port/endpoint/hsync-active=       00000000
| |-dcmi@4c006000/port/endpoint/vsync-active=       00000000
| |-dcmi@4c006000/port/endpoint/remote-endpoint=    0000003d
| |-dcmi@4c006000/port/endpoint/pclk-max-frequency= 0496ed40
| |-dcmi@4c006000/port/endpoint/bus-width=          00000008
| |-dcmi@4c006000/port/endpoint/pclk-sample=        00000000

| |-dcmi@4c006000/port/endpoint/pclk-max-frequency= 0496ed40 (77 MHz)
```

对于 HSYNC 和 VSYNC 信号，可通过 DCMI 状态寄存器（DCMI_SR）监控其引脚极性。此外，还可以检查这些引脚是否正确地连接到 DCMI 接口。

- 摄像头传感器 CSI 端口信号说明。

```
-[camera:ovti,ov5640]
| |-i2c@40013000/camera@3c/port/endpoint/data-lanes=
0000000100000002
| |-i2c@40013000/camera@3c/port/endpoint/clock-lanes=
00000000
| |-i2c@40013000/camera@3c/port/endpoint/pclk-max-frequency=
0496ed40
```

PWRDWN 和 RESET 信号极性设置

- 摄像头传感器 GPIO 信号极性说明。

```
…
-
[camera:ovti,ov5640]
| |-i2c@40013000/camera@3c/powerdown-gpios=    0000001d0000000500000000
| |-i2c@40013000/camera@3c/reset-gpios=
0000001c0000000c00000001
```

通过 GPIO 驱动复位（低电平有效）和掉电（高电平有效）引脚。对于 D3 Engineering 板上的 OV5640，GPIOA-12 (gpio-12) 和 GPIOB-5 (gpio-21) 分别用来在上电后释放传感器。

```
root@stm32mp1-av96:~# cat /sys/kernel/debug/gpio
gpiochip0: GPIOs 0-15, parent: platform/soc:pin-controller@50002000,
GPIOA:
 gpio-0   (                    |wakeup              ) in
hi
 gpio-12  (                    |reset               ) out
hi
gpiochip1: GPIOs 16-31, parent: platform/soc:pin-controller@50002000,
GPIOB:
 gpio-21  (                    |powerdown           ) out lo
```

对于 STMIPID02 复位（低电平有效），需是 GPIO 驱动的引脚。对于 D3 Engineering 板上的 Ov5640，使用 GPIOZ-0 (gpio-400)。

```
root@stm32mp1-av96:~# cat /sys/kernel/debug/gpio
…
gpiochip9: GPIOs 400-415, parent: platform/soc:pin-controller-z@54004000,
GPIOZ:
 gpio-400 (                    |reset               ) out hi
```

5.4 帧中断处理

DCMI 外设具有 DMA 请求功能，可以处理摄像头传感器的捕获帧，以便将其保存在存储器中。对于固定的帧格式，传感器可为每个帧发送相同数量的数据。然后，DMA 可以检测摄像头图像流中的每一帧，并发送传输结束中断，以通知 CPU 有一个帧等待处理。但是，这不适用于每帧数据量不确定的 JPEG 帧格式。在这种情况下，由 VSYNC 触发的 DCMI 帧结束中断将中止 DMA 传输，如下面的代码所示。

```
root@stm32mp1-av96:~# cat /proc/interrupts | grep dcmi
 52:          0          0     GIC-0 110 Level     4c006000.dcmi
```

5.5 帧格式的自定义

一些应用需要添加特定的帧编码格式（YUV、RGB 和 RAW）。OV5640 摄像头传感器内置集成图像传感器处理器，可对支持的帧格式和字节序列表中的内部数据格式进行转换。STMIPID02 和 DCMI 没有帧格式化功能，必须创建并在这些 Linux 驱动中相应地列出摄像头支持的所有格式。

在 OV5640 Linux 驱动程序中，提供了如下所示的支持格式列表（每个像素 8 位或 16 位）。如果应用中用到了这些格式，则 STMIPID02 和 DCMI 驱动也必须包含这些格式。

```
static const struct ov5640_pixfmt ov5640_formats[] = {
        { MEDIA_BUS_FMT_JPEG_1X8, V4L2_COLORSPACE_JPEG, },
        { MEDIA_BUS_FMT_UYVY8_2X8, V4L2_COLORSPACE_SRGB, },
        { MEDIA_BUS_FMT_YUYV8_2X8, V4L2_COLORSPACE_SRGB, },
        { MEDIA_BUS_FMT_RGB565_2X8_LE, V4L2_COLORSPACE_SRGB, },
        { MEDIA_BUS_FMT_RGB565_2X8_BE, V4L2_COLORSPACE_SRGB, },
        { MEDIA_BUS_FMT_SBGGR8_1X8, V4L2_COLORSPACE_SRGB, },
        { MEDIA_BUS_FMT_SGBRG8_1X8, V4L2_COLORSPACE_SRGB, },
        { MEDIA_BUS_FMT_SGRBG8_1X8, V4L2_COLORSPACE_SRGB, },
        { MEDIA_BUS_FMT_SRGGB8_1X8, V4L2_COLORSPACE_SRGB, },
};
```

5.6 摄像头传感器和 STMIPID02 输入时钟源

根据板设备树中的定义，在 DH Avenger96 板的设计中，OV5640 和 STMIPID02 输入时钟引脚均从单一时钟源信号驱动并连接到 STM32MP1 系列 MCO1 引脚。此方法省去了额外提供外部振荡器的成本。

MCO1 输出引脚从内部连接到 STM32MP1 系列外部高速振荡器（HSE 时钟源），提供了频率为 24 Mhz 的可靠时钟源。

为了将 MCO1 时钟驱动到摄像头传感器和 STMIPID02，只能进行一次引脚复用声明。摄像头传感器设备树节点（ov5640）必须托管时钟输入引脚复用定义，如以下代码所示。

```
ov5640: camera@3c {
         compatible = "ovti,ov5640";
         reg = <0x3c>;
+        pinctrl-names = "default", "sleep";
+        pinctrl-0 = <&rcc_pins_a>;
+        pinctrl-1 = <&rcc_sleep_pins_a>;

@@ -598,9 +601,6 @@
         compatible = "st,st-mipid02";
         reg = <0x14>;
         status = "okay";
-        pinctrl-names = "default", "sleep";
-        pinctrl-0 = <&rcc_pins_a>;
-        pinctrl-1 = <&rcc_sleep_pins_a>
```

5.7 设备树

STM32 微处理器设备树文件包含 STM32MP157 系列微处理器的硬件说明，其中包括 DCMI（stm32mp1 摄像头接口）设备树节点，stm32mp15c.dtsi 文件的位置如下：`<kernel_source_dir>/arch/arm/boot/dts`

stm32mp15c.dtsi 文件的说明信息中包含 STM32 DCMI 的硬件特性，这些特性是 STM32MP1 系列固有的，除非必须实现特定图像格式的要求，否则不得修改。

在更高层面，启动程序 U-boot 随内核映像一起加载板特定的已编译 DTS 文件（DTB），其中包括 DTSI 文件。STM32MP1 系列 I^2C 驱动程序通过 V4L2（video for Linux version 2）框架控制摄像头传感器设备。由于 I^2C 总线在运行时间没有动态可发现性来枚举设备（不同于 USB 协议等），因此在编译前，必须在板 DST 文件中充分描述摄像头设备属性。

OV5640 摄像头传感器设备由 STM32MP157 系列 I2C2 外设控制，任何新的摄像头传感器可能都一样。先描述 I2C2 总线属性，用于匹配摄像头设备规格。然后，它必须与 OV5640 摄像头传感器功率、时钟和板连接信号匹配，并最终与摄像头传感器和 STMIPID02 桥之间的端口连接匹配。提到的所有点都需要转移到新的摄像头传感器。设备树的详细信息如下。Avenger96 板的板设备树可在下面的位置找到：`<kernel-source>/arch/arm/boot/dts/stm32mp157a-av96.dts`

图 3. 设备树 - 第 1 部分

```
&i2c2 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&i2c2_pins_a>;
    pinctrl-1 = <&i2c2_pins_sleep_a>;
    i2c-scl-rising-time-ns = <185>;
    i2c-scl-falling-time-ns = <20>;
    status = "okay";
    /delete-property/dmas;
    /delete-property/dma-names;
```

STM32MP1 I2C2 properties

Binding of the device with the camera sensor driver (ov5640.c):

```
static const struct of_device_id ov5640_dt_ids[] = {
    { .compatible = "ovti,ov5640" },
}
```

```
    ov5640: camera@3c {
        compatible = "ovti, ov5640";
        reg = <0x3c>;
        pinctrl-names = "default", "sleep";
        pinctrl-0 = <&rcc_pins_a>;
        pinctrl-1 = <&rcc_sleep_pins_a>;
        clocks = <&rcc CK_MCO1>;
        clock-names = "xclk";
        assigned-clocks = <&rcc CK_MCO1>;
        assigned-clock-parents = <&rcc CK_HSE>;
        assigned-clock-rates = <24000000>;
        DOVDD-supply = <&v1v8>;
        reset-gpios = <&gpioa 12 GPIO_ACTIVE_LOW>;/* 31 PA12 TP54 - D3 mezz= GPIOI= TP6 CSI0_RESET (31 low speed) */    (1)
        powerdown-gpios = <&gpiob 5 GPIO_ACTIVE_HIGH>;/* 32 PB5 TP60 - D3 mezz= GPIOJ= TP7 CSI0_PWDN (32 low speed) */
        rotation = <180>;
        status = "okay";
```

I2C camera register address

camera sensor ov5640 power, clock and control signals

(1) *Here the sensor reset and power down signals are connected to the board GPIO expander and can be set active low or high depending of the sensor specifications.*

```
    port {
            ov5640_0: endpoint {
                remote-endpoint = <&mipid02_0>;
                clock-lanes = <0>;
                data-lanes = <1 2>;
                pclk-max-frequency = <77000000>;
            };
        };
    };
```

OV5640 MIPI CSI-2 data outputs signal connections to the STMIPID02 inputs.

STM32MP1 系列摄像头接口与 STMIPID02 解串行器桥之间的控制连接可以在该 DST 文件再往下的位置找到，并且不得更改。

图 4. 设备树 - 第 2 部分

```
&dcmi {
    status = "okay";
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&dcmi_pins_a>;
    pinctrl-1 = <&dcmi_sleep_pins_a>;

    port {
        dcmi_0: endpoint {
            remote-endpoint = <&mipid02_2>;
            bus-width = <8>;
            hsync-active = <0>;
            vsync-active = <0>;
            pclk-sample = <0>;
            pclk-max-frequency = <77000000>;
        };
    };
};
```

STMIPID02桥时钟和数据输出信号连接到STM32MP1摄像头接口(DCMI)输入。

像素时钟最大频率

STMIPID02 解串行器桥由 STM32MP1 系列 I2C4 外设控制，参考了桥两侧（STMIPID02 传感器和 STMIPID02-STM32MP1/DCMI）的连接。

图 5. 设备树 - 第 3 部分

```
&i2c4 {
…
    mipid02: mipid02@14 {
        compatible = "st,st-mipid02";
        reg = <0x14>;
        status = "okay";
        clocks = <&rcc CK_MCO1>;
        clock-names = "xclk";
        assigned-clocks = <&rcc CK_MCO1>;
        assigned-clock-parents = <&rcc CK_HSE>;
        assigned-clock-rates = <24000000>;
        VDDE-supply = <&v1v8>;
        VDDIN-supply = <&v1v8>;
        reset-gpios = <&gpioz 0 GPIO_ACTIVE_LOW>;
        ports {
            #address-cells = <1>;
            #size-cells = <0>;
            port@0 {
                reg = <0>;

                mipid02_0: endpoint {
                    data-lanes = <1 2>;
                    remote-endpoint = <&ov5640_0>;
                };
            };
            port@2 {
                reg = <2>;

                mipid02_2: endpoint {
                    bus-width = <8>;
                    hsync-active = <0>;
                    vsync-active = <0>;
                    pclk-sample = <0>;
                    remote-endpoint = <&dcmi_0>;
                };
            };
        };
    };
```

STMIPID02 桥的功率、时钟和控制信号。

STMIPID02 <-> 摄像头模块数据连接。

STMIPID02数据输出信号 <–> STM32MP1摄像头接口（DCMI）数据输入和活动状态规格。

5.8 检查 GPIO 引脚连接

在完成连接后，必须从 STM32MP1 系列主机的角度，检查通过 STMIPID02 传播的所有摄像头传感器信号是否配置正确。

5.8.1 STM32MP1 系列摄像头接口 GPIO 设置

使用 CubeIDE 工具可以轻松地完成设置。意法半导体的图形界面工具可以配置所有 STM32MP1 系列产品需要的 GPIO，例如下面的 DCMI 接口。

```
 root@stm32mp1-av96:~# cat /sys/kernel/debug/pinctrl/soc\:pin-controller*/pin* | grep dcmi
pin 4 (PA4): device 4c006000.dcmi function af13 group PA4
pin 6 (PA6): device 4c006000.dcmi function af13 group PA6
pin 9 (PA9): device 4c006000.dcmi function af13 group PA9
pin 23 (PB7): device 4c006000.dcmi function af13 group PB7
pin 64 (PE0): device 4c006000.dcmi function af13 group PE0
pin 65 (PE1): device 4c006000.dcmi function af13 group PE1
pin 70 (PE6): device 4c006000.dcmi function af13 group PE6
pin 119 (PH7): device 4c006000.dcmi function af13 group PH7
pin 122 (PH10): device 4c006000.dcmi function af13 group PH10
pin 126 (PH14): device 4c006000.dcmi function af13 group PH14
pin 129 (PI1): device 4c006000.dcmi function af13 group PI1
pin 132 (PI4): device 4c006000.dcmi function af13 group PI4
pin 134 (PI6): device 4c006000.dcmi function af13 group PI6
```

对于 GPIO 引脚和相关功能（AF13），请参见 STM32MP1 系列数据手册中的特定 DCMI 信号。 表 2. 参考文档

表 4. GPIO 引脚和相关功能

STM32MP1 系列 GPIO 引脚	STM32MP1 系列复用功能（AF13）
PA4	DCMI_HSYNC
PA6	DCMI_PIXCLK
PA9	DCMI_D0
PB7	DCMI_VSYNC
PE0	DCMI_D2
PE1	DCMI_D3
PE6	DCMI_D7
PH7	DCMI_D9
PH10	DCMI_D1
PH14	DCMI_D4
PI1	DCMI_D8
PI4	DCMI_D5
PI6	DCMI_D6

提示 *10 位数据以物理方式连接到 DCMI 外设接口，STMIPID02 和 DCMI 驱动以 MIPI CSI-2 模式高效处理 8 位数据 DCMI[7:0]。这是根据选定图像格式的 STMIPID02 规格，在 STMIPID02 寄存器设置（Mode_Reg1 bit7）中处理的。*

5.9 调试和跟踪指令

本节将提供一些实用的 Linux 控制台指令，可用于查看传感器或确保 Linux 内核是否正确加载了摄像头器件驱动。

5.9.1 跟踪指令

为了进行跟踪，必须使用 dmesg 指令打印（或控制）内核环形缓冲区。在默认情况下，只跟踪错误和警告。将细节的粒度级别更改为最大值（8 级），可以将所有跟踪数据同步输出到控制台。

指令$> dmesg -n8 打印控制台中的所有跟踪调试数据（dev_dbg）。注意，该指令会显示大量跟踪数据。建议只进行选择性的搜索。该指令只跟踪与摄像头传感器 ov5640 相关的内核信息。

```
root@stm32mp1-av96:~# dmesg | grep ov5640
[    8.329669] ov5640 0-003c: Linked as a consumer to regulator.15
[    8.329718] ov5640 0-003c: 0-003c supply DVDD not found, using dummy regulator
[    8.329818] ov5640 0-003c: Linked as a consumer to regulator.0
[    8.329846] ov5640 0-003c: 0-003c supply AVDD not found, using dummy regulator
```

5.9.2 摄像头模块探查

在 Linux 启动后，使用 rmmod 和 modprobe 指令分别可以解绑和绑定器件驱动。这些指令可用于检查器件的当前状态，以及 Linux 内核是否正确加载了驱动。此外，日志错误会提示问题的源头。例如，在引脚复用 GPIO 复用功能与共享同一引脚的另一个外设发生冲突时。

modeprobe 还可以检索产品器件 ID，并提供与该器件通信时使用的外设的详细信息。它还能列出与摄像头模块相连的所有附属模块。

```
root@stm32mp1-av96:~#modprobe -D ov5640
insmod /lib/modules/4.19.49/kernel/drivers/media/media.ko
insmod /lib/modules/4.19.49/kernel/drivers/media/v4l2-core/videodev.ko
insmod /lib/modules/4.19.49/kernel/drivers/media/v4l2-core/v4l2-common.ko
insmod /lib/modules/4.19.49/kernel/drivers/media/v4l2-core/v4l2-fwnode.ko
insmod /lib/modules/4.19.49/kernel/drivers/media/i2c/ov5640.ko
```

5.9.3 动态调试

执行动态调试可以从特定驱动动态地检索内核信息（如 DCMI dev_dbg 信息），无需为模块打补丁和进行重新编译。解绑/绑定 DCMI 驱动是 rmmod/modprobe 指令的替代选择，可以在维持该模块的动态调试配置激活状态的同时，从最终用户空间将驱动从器件上手动断开。此探头输出调试跟踪信息。

```
root@stm32mp1-av96:~# dmesg -n8
root@stm32mp1-av96:~# echo "module stm32_dcmi +p" > /sys/kernel/debug/dynamic_debug/control
root@stm32mp1-av96:~# echo -n "4c006000.dcmi" >  /sys/bus/platform/drivers/stm32-dcmi/unbind;
echo -n "4c006000.dcmi" >  /sys/bus/platform/drivers/stm32-dcmi/bind
[ 5431.821221] stm32-dcmi 4c006000.dcmi: Removing video0
[ 5431.830087] stm32-dcmi 4c006000.dcmi: Device registered as video0
[ 5431.841444] stm32-dcmi 4c006000.dcmi: Subdev "st-mipid02 2-0014" bound
[ 5431.849759] stm32-dcmi 4c006000.dcmi: DCMI is now linked to "st-mipid02 2-0014"
[ 5431.858627] stm32-dcmi 4c006000.dcmi: Supported fourcc/code: RGBP/0x1008
[ 5431.866478] stm32-dcmi 4c006000.dcmi: Supported fourcc/code: YUYV/0x2008
[ 5431.871725] stm32-dcmi 4c006000.dcmi: Supported fourcc/code: UYVY/0x2006
[ 5431.882724] stm32-dcmi 4c006000.dcmi: Supported fourcc/code: JPEG/0x4001
[ 5431.895508] stm32-dcmi 4c006000.dcmi: Probe done
```

动态调试跟踪能够监控关联到器件总线标识符的 stm32-dcmi 驱动绑定。此外，还能检索 DCMI 驱动支持的 Fourcc 格式标识符，并确保 STMIPID02 作为子器件正确地绑定到 DCMI。

root@stm32mp1-av96:~#echo "module st_mipid02 +p" > /sys/kernel/debug/dynamic_debug/control

```
[ 1244.381391] st-mipid02 2-0014: mipid02_set_fmt for 0
[ 1244.444326] st-mipid02 2-0014: mipid02_get_fmt probe 0
[ 1244.448281] st-mipid02 2-0014: mipid02_s_stream : requested 1 / current = 0
[ 1244.455136] st-mipid02 2-0014: detect link_freq = 384000000 Hz
[ 1244.465744] st-mipid02 2-0014: mipid02_s_stream current now = 1 / 0
```

5.9.4 检查帧率

为了检查 v4l2 框架生成、测量各图像的频率与传感器帧率指令之间是否存在差异，可运行下列 v4l2-ctl 指令。

```
root@stm32mp1-av96:~# v4l2-ctl --set-parm=15;

root@stm32mp1-av96:~# v4l2-ctl --set-fmt-video=width=1280,height=720,pixelformat=RGBP --
stream-mmap --stream-count=-1 &

Frame rate set to 15.000 fps

<<<<<<<<<<<<<<<<<< 15.14 fps
<<<<<<<<<<<<<<<<<< 15.14 fps
…
```

具体情形如下：

- 期望帧率

```
root@stm32mp1-av96:~# v4l2-ctl --set-parm=15;
```

- 实测帧率

```
<<<<<<<<<<<<<<<<<< 15.14 fps
<<<<<<<<<<<<<<<<<< 15.14 fps
…
```

5.9.5 比较 jpg 文件

下面是 v4l2 和 GStreamer 对 jpg 文件进行二进制比较的结果（两张测试模式图像和一个捕获帧）。

```
root@stm32mp1-av96:~# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=JPEG --set-
ctrl=test_pattern=1 --set-parm=30 --stream-mmap --stream-count=1 --stream-to=sanity-
colorbars-0.jpeg
Frame rate set to 30.000 fps
<
root@stm32mp1-av96:~# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=JPEG --set-
ctrl=test_pattern=1 --set-parm=30 --stream-mmap --stream-count=1 --stream-to=sanity-
colorbars-1.jpeg
Frame rate set to 30.000 fps
<
root@stm32mp1-av96:~# v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=JPEG --set-
ctrl=test_pattern=0 --set-parm=30 --stream-mmap --stream-count=1 --stream-to=sanity-
nocolorbars.jpeg
Frame rate set to 30.000 fps
<
root@stm32mp1-av96:~# echo "**** Camera sanity check ****" >> camerasanity.txt
cho root@stm32mp1-av96:~# echo "Similarities between 2 consecutives pictures (should be >
0.9):" >> camerasanity.txt
root@stm32mp1-av96:~# gst-launch-1.0 filesrc location= sanity-colorbars-0.jpeg ! decodebin !
compare name=cmp method=ssm
 meta=none threshold=0  ! fakesink  filesrc location= sanity-colorbars-1.jpeg ! decodebin !
cmp. -v --gst-debug=*BUS*:5 2
>&1 | grep "content=" | grep dispatch  | awk -F")" '{print $(NF)}' >> camerasanity.txt
root@stm32mp1-av96:~# echo "Similarities with control picture (should be < 0.7):" >>
camerasanity.txt
root@stm32mp1-av96:~# gst-launch-1.0 filesrc location= sanity-colorbars-0.jpeg ! decodebin !
compare name=cmp method=ssim
 meta=none threshold=0  ! fakesink  filesrc location= sanity-nocolorbars.jpeg ! decodebin !
cmp. -v --gst-debug=*BUS*:5 2
>&1 | grep "content=" | grep dispatch | awk -F")" '{print $(NF)}' >> camerasanity.txt
root@stm32mp1-av96:~# weston-image sanity-colorbars-0.jpeg sanity-colorbars-1.jpeg sanity-
nocolorbars.jpeg &

root@stm32mp1-av96:~# sleep 5
root@stm32mp1-av96:~# kill %1
root@stm32mp1-av96:~# cat camerasanity.txt
**** Camera sanity check ****
Similarities between 2 consecutives pictures (should be > 0.9):
1;
Similarities with control picture (should be < 0.7):
0.63874228088429863;
```

版本历史

表 5. 文档版本历史

日期	版本	变更
2020 年 9 月 3 日	1	初始版本

目录

表一览

图一览