STM32MP13x 产品与 MIPI® CSI-2 摄像头对接

简介

本文档提供关于如何将 STM32MP13x 系列产品与 MIPI CSI-2 摄像头连接的信息。STM32MP13x 系列可以通过 DCMIPP（数码摄像头模块接口像素处理器）并行端口对 CMOS 摄像头传感器进行寻址。但是，得益于 STMIPID02 MIPI CSI-2 解串行器分立器件，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）。

MIPI CSI-2 接口协议多年以来一直是当今嵌入式传感器的标准技术。这是由移动应用市场推动的，也广泛应用于工业市场。与传统的并行接口或 MIPI CPI 相比，MIPI CSI-2 具有决定性的优势，其引脚数量更少且成本更低。

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）。

本应用笔记的目的是借助 STM32MP135F-DK 板演示 STM32MP13x 系列通过 STMIPID02 MIPI CSI-2 解串行器满足 200 万像素 GC2145 MIPI CSI-2 摄像头传感器的应用需求能力。这两种驱动都已实现，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

本文档适用于基于 STM32MP13xx Arm® Cortex®的微处理器。

注意 *Arm 是 Arm Limited（或其子公司）在美国和/或其他地区的注册商标。*

arm

表 1. 缩略语列表

缩略语	说明
CPI	摄像头并行接口
CSI	摄像头串行接口
DCMIPP	数码摄像头接口像素处理器
LDO	低压降调节器
MIPI	移动产业处理器接口
PMIC	电源管理芯片

2 参考文档

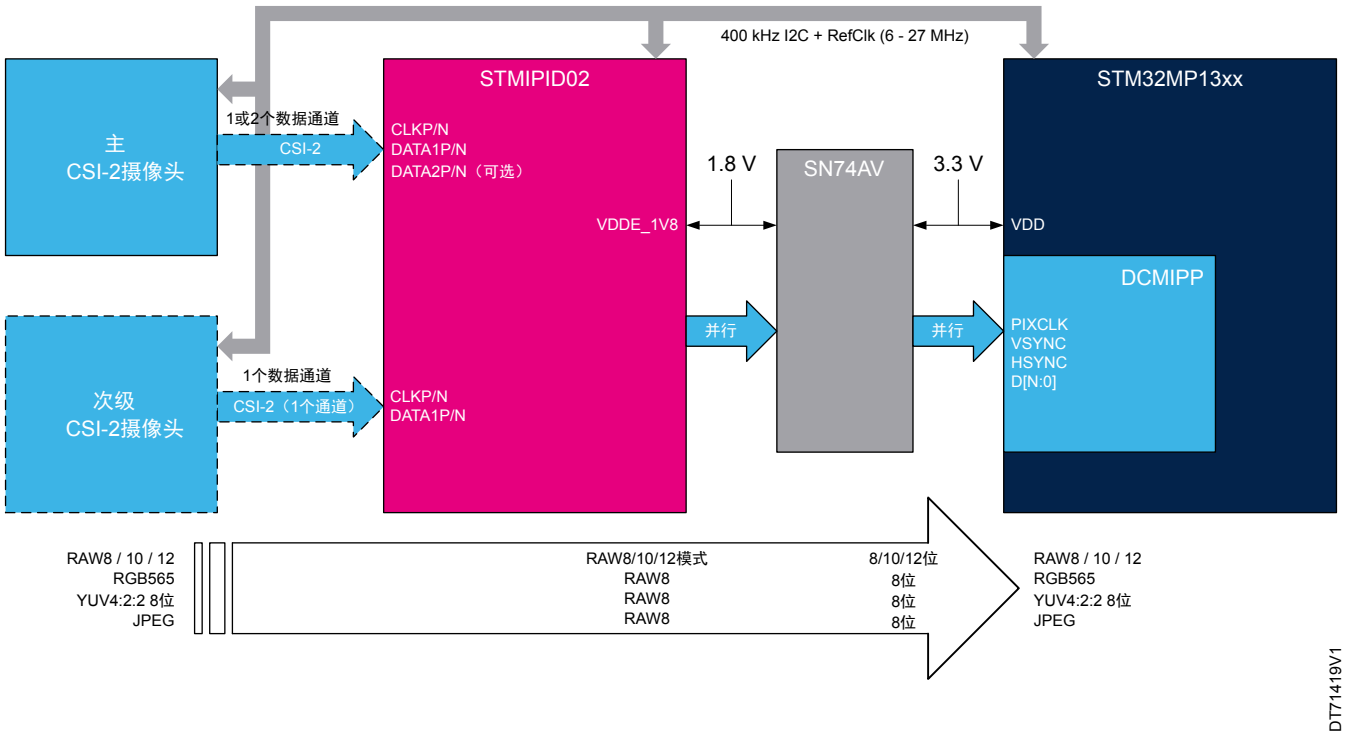
下面的资源是公开的，可以在意法半导体网站 *www.st.com* 找到。

参考编号	文件标题
[R1]	STM32MP13x 数据手册：DS13483、DS13874、DS13875、DS13876、DS13877、DS13878。
[R2]	STM32MP131x/3x/5x 器件勘误表：ES0539。
[R3]	双模式 MIPI CSI-2 / SMIA CCP2 解串行器：DS12803
[R4]	STM32MP13x 系列硬件开发入门：AN5474
[R5]	Linux® V4L2 摄像头框架面向 STM32MP13：https://wiki.st.com/stm32mpu/ + 搜索 'STM32MP13 V4L2 camera overview'。
[R6]	STCubeProgrammer（Flash 编程工具）：https://wiki.st.com/stm32mpu/wiki/STM32CubeProgrammer。
[R7]	设备树配置：https://wiki.st.com/stm32mpu/ + 搜索'DCMIPP device tree configuration'。
[R8]	OpenSTLinux_distribution：https://wiki.st.com/stm32mpu/wiki/OpenSTLinux_distribution#Reference_source_code。
[R9]	OpenSTLinux 发行软件包的目录结构：https://wiki.st.com/stm32mpu/wiki/Example_of_directory_structure_for_Packages。
[R10]	STM32CubeProgrammer 软件工具::https://www.st.com/en/development-tools/stm32cubeprog.html。
[R11]	https://wiki.st.com/stm32mpu/wiki/how_to_populate_the_sd_card_with_dd_command.
[R12]	https://wiki.st.com/stm32mpu/wiki/How_to_use_USB_mass_storage_in_U-Boot.
[R13]	https://wiki.st.com/stm32mpu/wiki/STM32MP13_resources#Boards_user_manuals.
[R14]	STM32MP15x 序列产品连接 MIPI CSI-2 摄像头：AN5470。

3 STM32MP13x 系列产品与 STMIPID02 MIPI CSI-2 解串行器的接口连接

STM32MP13xx 微处理器没有原生集成 MIPI CSI-2 接口，而是内置基于 MIPI CPI接口的 DCMIPP 并行端口。可通过 STMIPID02 MIPI CSI-2 解串行器对其进行连接，以对接任何兼容的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 STM32MP13xx 微处理器 DCMIPP 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

与 MIPI CPI 相比，MIPI CSI-2 接口的引脚数量明显减少。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

由于 STMIPID02 解串行器桥外部电源引脚的电压被限制为 1.8 V，STM32MP135F-DK 板内嵌 1.8 V-3.3 V 电平转换器（SN74AV），为 STM32MP135F 提供 V_{DD} = 3.3 V 标称电压。面向 STM32MP135F 的所有不同电压都是通过外部 PMIC 模块（电源管理芯片）提供的。

如需详细了解整体原理，详见 STM32MP135F-DK 母板原理图[R13]。如需将 STM32MP13F 配置为 V_{DD} = 1.8 V 电压，请参见[R4]。

对于 GC2145 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 STM32MP13x 系列产品通过 DCMIPP 实现的视频吞吐率性能

MIPI CSI-2.1 接口理论上可以通过 D-PHY 支持每通道最高 2.5 Gbyte/s 的数据吞吐速率。而 STM32MP13xx 不能持续做到这一点，首先是由于（MIPI CPI）并行端口接口的引脚上有 I/O 转换速率限制，其次是因为 MPU 很难实时处理摄像头源源不断发来的帧率。

例如，2-Mpixel 传感器的每像素位数为 16 位，帧率为 30 帧/s，得出连续数据吞吐率为 120 Mbyte/s。这样的数据在并行端口接口上几乎是不可持续的。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 GC2145 传感器到 STM32MP13xx 微控制器，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：

- VGA 640 x 480 RGB 565 30 fps
- 720p 1280 x 720 RGB 565 30fps
- 720p 1280 x 720 YUYV 30fps
- UXGA 1600 x 1200 RGB 565 20fps
- UXGA 1600 x 1200 YUYV 20fps

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。为了满足借助人工智能技术将这类传感器从工业市场转向物联网（IoT）领域这一日益增长的需求，STMIPID02 驱动程序已经上传到 Linux 社区。在基于 Linux 的应用中可以免费获取它。STMIPID02 桥驱动器包含在 STMicroelectronics OpenSTLinux 软件交付包中（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。它被移植到具有 STM32MP135F 和 STMIPID02 的 STM32MP135F- DK 探索板上，通过 STM32MP135F- DK 板级支持包中的带状电缆与具有 GC2145 摄像头传感器的 MB1897 摄像头板件模块连接。

图 2. 摄像头演示程序硬件

4.1 STM32MP135F-DK 探索板

该板主要包括：

- STM32MP135F 11x11 微处理器
- STPMIC1D 电源模块
- 2 × 512 MB 的 DDR3L RAM
- STMIPID02 解串行器桥
- 通过 USB Micro-B 端口的 STLINK-V3E（将控制台连接到主机 PC）
- 2× USB TypeC，用于实现 flash 加载和 DRP（通过 STM32G0 MCU）。
- 连接，用于接收 MB1897 GC2145 摄像头板件模块数据

若需更多信息，请参考[R13]。

4.2 MB1897 摄像头板模块概述

GC2145 摄像头传感器焊接在 MB1897 电路板上，通过 MIPI CSI-2 连接到 STM32MP135F，用于评估目的。

4.3 板级镜像

意法半导体的 OpenSTLinux 软件针对 STM32MP 应用板（包括 STM32MP135F-DK 板），通过其发行软件包提供驱动程序、库、工具和示例来运行 STM32MP13x 系列嵌入式外设。它本身通过 STMIPID02 MIPI CSI-2 解串行器桥对 MIPI CSI-2 摄像头传感器（如 GC2145）进行寻址。

至于演示器，OpenSTLinux Starter 软件包提供了准备在 STM32MP135F-DK 板 SDcard 内存中刷写的板级镜像文件，并通过 STMIPID02 解串行器桥嵌入 GC2145 传感器或 OV5640 摄像头传感器（现已弃用）的驱动程序。

下一节将详细介绍推荐的两种镜像编程过程。

4.3.1 STM32CubeProgrammer 软件工具

利用该一体化软件工具，用户可以轻松将二进制镜像烧录到 STM32 MCU/MPU 上。该工具可以从[R10]下载。

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

表 2. 面向编程的启动开关配置

启动模式	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	0	0	0

一旦二进制镜像烧录完毕，将硬件启动开关选为 SDCard 启动模式。

1. Boot2 = 1
2. Boot1 = 0
3. Boot0 = 1

建议将整个二进制镜像烧录到一个 SDcard 中。

对于定制板件，如果二进制镜像分区必须在 NOR-Flash（TF-A、U-boot）和 e•MMC（Linux）之间完成，请参见[R6]。

4.3.2 其他编程方法

这里有两种其他编程方法：

- 要使用传统“dd”命令将二进制镜像烧录到 SDcard 上，请参阅[R11]指南。
- 要使用 U-boot 将二进制镜像烧录到 USB 大容量存储设备上，请参阅[R12]。

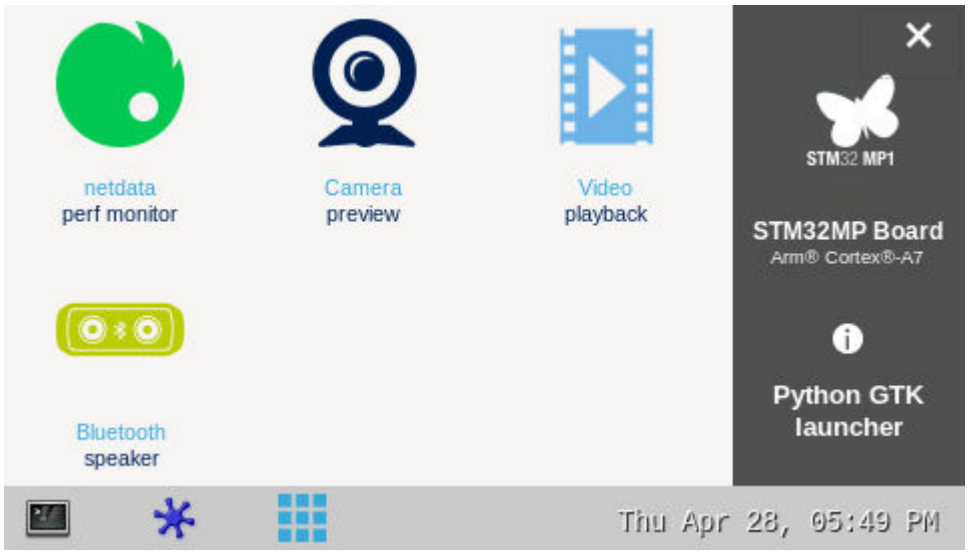
注意 如果有 U-boot 可用，从而 SDCard 或板的非易失性存储器上已有两个原始 Flash 板分区，则适用此方法。

4.4 启动板级镜像和激活摄像头预览屏幕

一旦启动开关改为 SDcard：

- GTK 启动程序演示器屏幕应弹出（参见图 3）。
- 选择摄像头预览图标，显示来自 GC2145 MIPI CSI-2 摄像头模块的实时视频流。

图 3. GTK 启动程序演示器



用于运行摄像头预览的脚本位于*/usr/local/demo/application/camera/bin/launch_camera_preview.sh*。默认情况下，传感器配置为 VGA RGB565 30fps，然后按比例缩小数据流以适应板件屏幕分辨率（480 x 272）。

摄像头传感器格式和帧率设置通过 STM32MP13xx 微处理器和 GC2145 摄像头传感器专用的 V4l2 Linux 内核框架进行调整。下节将提供一个命令子集。有关摄像头子系统拓扑结构和设置的全面信息，请参见[R5]。

4.5 V4L2 指令

在 STM32MP13x 上，必须首先配置摄像头子系统，然后调用 V4L2 应用相关的指令。这是通过 V4L2 `media-ctl` 实用命令执行的（参加[R6]）。

注意 *上述步骤具有 STM32MP13x 专一性。在 STM32MP15x 产品上，V4l2 指令可以通过 I2C 直接与摄像头传感器对接（参加第 2 节: [R15]）。*

4.5.1 配置摄像头子节点和帧捕获

下面的 shell 脚本可用于通过以下格式配置 GC2145：RGB565_BE、RGB565_LE、YVYU、YUYV、UYVY、VYUY。BE/LE 确定数据字节存储次序（仅用于 RGB565 格式）。

Media-ctl 部分配置摄像头子系统，而 v4l2 命令执行视频捕获任务：

```
if `echo $3 | grep "RGB565_LE" 1>/dev/null 2>&1`; then
  sensorbuscode=RGB565_2X8_LE
  parallelbuscode=RGB565_2X8_LE
  v4l2ctlfmt=RGBP
fi
if `echo $3 | grep "RGB565_BE" 1>/dev/null 2>&1`; then
  sensorbuscode=RGB565_2X8_BE
  parallelbuscode=RGB565_2X8_LE
  v4l2ctlfmt=RGBP
fi
if `echo $3 | grep "YUYV" 1>/dev/null 2>&1`; then
  sensorbuscode=YUYV8_2X8
  parallelbuscode=YUYV8_2X8
  v4l2ctlfmt=YUYV
fi
if `echo $3 | grep "YVYU" 1>/dev/null 2>&1`; then
  sensorbuscode=YVYU8_2X8
  parallelbuscode=YUYV8_2X8
  v4l2ctlfmt=YVYU
fi
if `echo $3 | grep "UYVY" 1>/dev/null 2>&1`; then
  sensorbuscode=UYVY8_2X8
  parallelbuscode=UYVY8_2X8
  v4l2ctlfmt=UYVY
fi
if `echo $3 | grep "VYUY" 1>/dev/null 2>&1`; then
  sensorbuscode=VYUY8_2X8
  parallelbuscode=VYUY8_2X8
  v4l2ctlfmt=VYUY
fi

media-ctl -d /dev/media0 --set-v4l2 "'gc2145 1-003c':0[fmt:$sensorbuscode/$1x$2@1/$4 field:no
ne]" -v
media-ctl -d /dev/media0 --set-v4l2 "'dcmipp_parallel':0[fmt:$sensorbuscode/$1x$2]" -v
media-ctl -d /dev/media0 --set-v4l2 "'dcmipp_parallel':1[fmt:$parallelbuscode/$1x$2]" -v
media-ctl -d /dev/media0 --set-v4l2 "'dcmipp_dump_postproc':1[fmt:$parallelbuscode/$1x$2]" -v
media-ctl -d /dev/media0 --set-v4l2 "'dcmipp_dump_postproc':1[crop:(0,0)/$1x$2]" -v
v4l2-ctl --set-fmt-video=width=$1,height=$2,pixelformat=$v4l2ctlfmt --stream-mmap --stream-sk
ip=100 --stream-count=1 --stream-to=my_image
```

即将传感器配置为 VGA 格式，视频格式为 RGB565_BE，目标帧速率为 30fps:

```
root@stm32mp1:~# ./capture_test_gc2145.sh 640 480 RGB565_BE 30
Opening media device /dev/media0
Enumerating entities
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
Found 5 entities
Enumerating pads and links
Setting up format RGB565_2X8_BE 640x480 on pad gc2145 1-003c/0
Format set: RGB565_2X8_BE 640x480
Setting up frame interval 1/30 on pad gc2145 1-003c/0
Frame interval set: 1/30
Setting up format RGB565_2X8_BE 640x480 on pad st-mipid02 1-0014/0
Format set: RGB565_2X8_BE 640x480
Setting up frame interval 1/30 on pad st-mipid02 1-0014/0
Enumerating entities
looking up device: 81:1
```

```
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
Found 5 entities
Enumerating pads and links
Setting up format RGB565_2X8_BE 640x480 on pad dcmipp_parallel/0
Format set: RGB565_2X8_BE 640x480
Opening media device /dev/media0
Enumerating entities
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
Found 5 entities
Enumerating pads and links
Setting up format RGB565_2X8_LE 640x480 on pad dcmipp_parallel/0
Format set: RGB565_2X8_BE 640x480
Setting up format RGB565_2X8_LE 640x480 on pad dcmipp_dump_postproc/0
Format set: RGB565_2X8_BE 640x480
Opening media device /dev/media0
Enumerating entities
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
Found 5 entities
Enumerating pads and links
Setting up format RGB565_2X8_LE 640x480 on pad dcmipp_dump_postproc/0
Format set: RGB565_2X8_BE 640x480
Opening media device /dev/media0
Enumerating entities
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
looking up device: 81:1
Found 5 entities
Enumerating pads and links
Setting up selection target 0 rectangle (0,0)/640x480 on pad dcmipp_dump_postproc/1
Selection rectangle set: (0,0)/640x480
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 29.03 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 29.03 fps
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< 29.03 fps
<<<<<<<<<<<<
```

4.5.2 在 Weston/Wayland 中显示摄像头预览

默认情况下，用户以根用户身份登录。从 OpenSTLinux DV4.1 起，如要启动 Weston Wayland 应用程序并运行 weston/wayland 命令，需要以 weston 用户身份登录：

```
root@stm32mp1:~# su -l weston
stm32mp1:~$
```

摄像头子系统必须按照所选的格式、分辨率和帧速率进行设置（参见 4.5.1）。例如，可以在 gstreamer 通道中调用 waylandsink，使用与摄像头子系统设置相同的格式：

```
stm32mp1:~$ ./media-ctl-setup_gc2145.sh 640 480 RGB_565_BE 30
stm32mp1:~$ gst-launch-1.0 v4l2src device=/dev/video0 ! "video/x-raw,width=640,height=480,fra
merate=30/1" ! waylandsink fullscreen=true
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Redistribute latency...
```

来自 GC2145 摄像头的图像预览显示在 STM32MP13F-DK 板件的 LCD 屏幕上。

修订历史

表 3. 文档修订历史

日期	版本	变更
2023 年 1 月 31 日	1	初始版本。

目录

表一览

图一览