AN5478

Application note

STM32MP13x lines interfacing with a MIPI® CSI-2 camera

Introduction

The document provides information on how to interface STM32MP13x line microprocessors with a MIPI CSI-2 camera. The STM32MP13x lines can address CMOS camera sensors through the DCMIPP (digital camera module interface pixel processor) parallel port. However, it is possible to extend the range of addressable camera sensors, for instance MIPI®CSI-2 cameras (camera serial interface), thanks to the STMIPID02 MIPI CSI-2 deserializer discrete component.

The MIPI CSI-2 interface protocol has become for many years the standard technology for today's embedded sensors. This has been driven by the mobile market and is also widely used in the industrial market. MIPI CSI-2 brings decisive advantages, offering reduced pin count and lower cost versus the conventional parallel interface or MIPI CPI.

The STMIPID02 MIPI CSI-2 deserializer can address a broad range of MIPI CSI-2 camera sensors used in mobile devices and automotive applications. The direct interface removes the requirement for associated software overhead linked to frame decoding (for camera over USB or Ethernet for instance).

Relying on the STM32MP135F-DK board, the purpose of this application note is to demonstrate the STM32MP13x lines capability to address the 2 Mpixel GC2145 MIPI CSI-2 camera sensor through the STMIPID02 MIPI CSI-2 deserializer. Both drivers are available and included in the STMicroelectronics OpenSTLinux software distribution package. For this application note and according to the STMIPID02 deserializer specifications, the focus is set exclusively on the MIPI CSI-2.1 protocol over a D-PHY.

1 General information

This document applies to STM32MP13xx Arm® Cortex® based microprocessors.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Table 1. List of acronyms

Acronym	Description
CPI	Camera parallel interface
CSI	Camera serial interface
DCMIPP	Digital camera interface pixel processor
LDO	Low-dropout regulator
MIPI	Mobile industry processor interface
PMIC	Power management integrated circuit

2 Reference documents

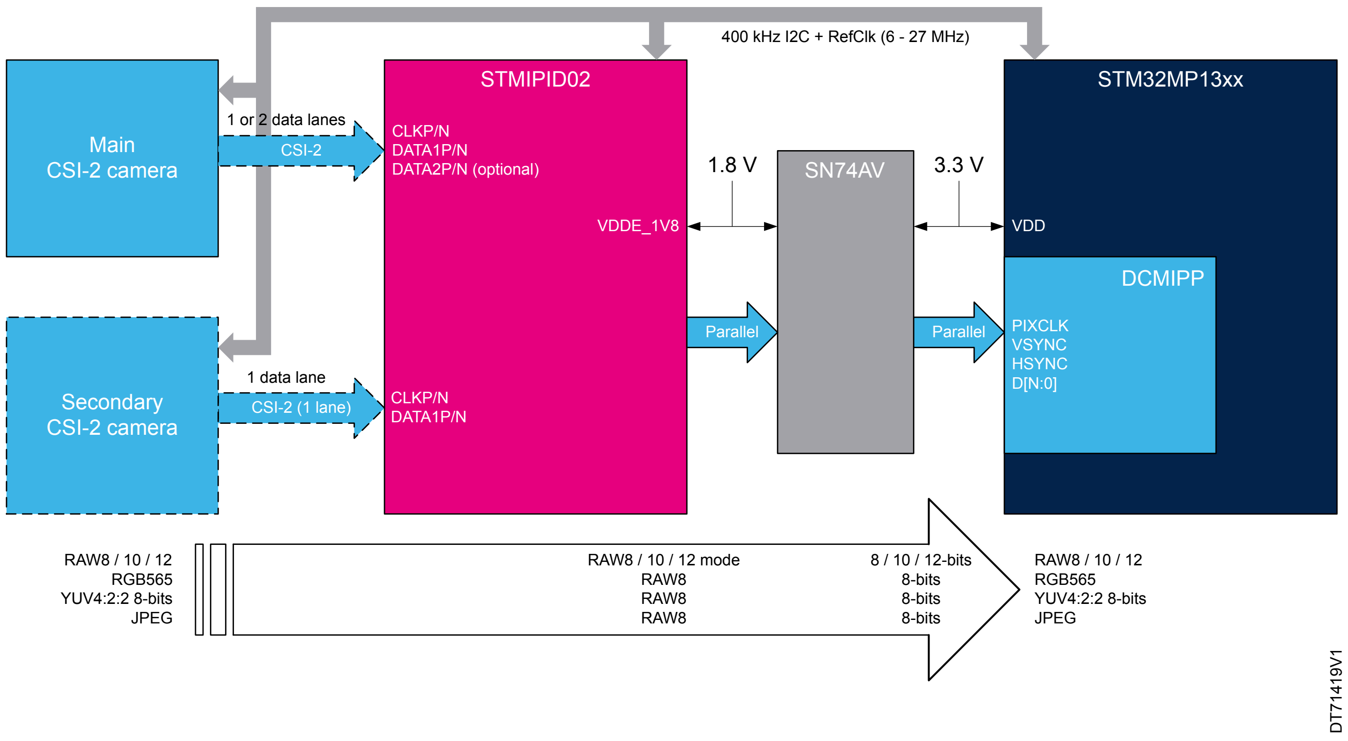
The below resources are public and available on STMicroelectronics web site at *www.st.com*.

Reference number	Document title
[R1]	STM32MP13x datasheets: DS13483, DS13874, DS13875, DS13876, DS13877, DS13878.
[R2]	STM32MP131x/3x/5x device errata: ES0539.
[R3]	Dual mode MIPI CSI-2 / SMIA CCP2 de-serializer: DS12803.
[R4]	Getting started with STM32MP13x lines hardware development: AN5474.
[R5]	Linux® V4L2 camera framework for STM32MP13: https://wiki.st.com/stm32mpu/ + search 'STM32MP13 V4L2 camera overview'.
[R6]	STCubeProgrammer (flash programming tool): https://wiki.st.com/stm32mpu/wiki/STM32CubeProgrammer.
[R7]	Device tree configuration: https://wiki.st.com/stm32mpu/ + search 'DCMIPP device tree configuration'.
[R8]	OpenSTLinux_distribution: https://wiki.st.com/stm32mpu/wiki/OpenSTLinux_distribution#Reference_source_code.
[R9]	Directory structure for the OpenSTLinux Distribution package: https://wiki.st.com/stm32mpu/wiki/Example_of_directory_structure_for_Packages.
[R10]	STM32CubeProgrammer software tool:: 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 Series interfacing with a MIPI CSI-2 camera: AN5470.

3 STM32MP13x lines interfacing with the STMIPID02 MIPI CSI-2 deserializer

The STM32MP13xx microprocessors do not natively implement a MIPI CSI-2 interface but embed a DCMIPP parallel port based on a MIPI CPI interface. It can be connected through the STMIPID02 MIPI CSI-2 deserializer to address any compatible MIPI CSI-2 camera sensor device. The STMIPID02 MIPI CSI-2 deserializer is connected to a MIPI CSI-2 camera on one side, and to the STM32MP13xx microprocessor DCMIPP data parallel interface on the other side. An overview of the block diagram is shown in the figure below.

Figure 1. Block diagram overview



3.1 MIPI CSI-2 versus MIPI CPI interface

Compared to a MIPI CPI, the MIPI CSI-2 interface offers a very significant pin count saving. A MIPI CPI data port requires a minimum of eight data lines (12 data lines maximum), one clock, two synchronization lines, where a MIPI CSI-2 data port requires a 2-wire differential pair per lane, and a clock lane.

3.2 Power supply considerations

Since the STMIPID02 deserializer bridge external power-supply voltage pins are limited to 1.8 V, the STM32MP135F-DK board embeds 1.8 V-3.3 V level shifters (SN74AV) to fed the STM32MP135F at the nominal V_{DD} = 3.3 V. All the different power voltages to the STM32MP135F are supplied through the external PMIC module (power management integrated circuit).

For detailed information about the overall schematics refer to the STM32MP135F-DK motherboard schematics [R13]. To configure the STM32MP13F with V_{DD} = 1.8 V supply voltage, refer to [R4].

Regarding the GC2145 camera sensor, the V_{DD} power supply for the I/Os and the LDO (low-dropout regulator) external power source is set to 1.8 V. For the analog logic, a 2.8 V must also be supplied as well as the external source.

3.3 STM32MP13x lines video throughput performance through DCMIPP

The MIPI CSI-2.1 interface can theoretically support data throughput rates up to 2.5 Gbyte/s per lane with a D-PHY. This is hardly sustainable by the STM32MP13xx, firstly due to I/O slew rate constraints on the pins of the (MIPI CPI) parallel port interface, secondly because the MPU would have hard time processing in real time such a frame rate coming continuously from the camera.

For instance, a 2-Mpixel sensor with 16-bit per pixel and a frame rate of 30 frames/s, produces a continuous data throughput of 120 Mbyte/s. Such figures are hardly sustainable on a parallel port interface. Therefore, the sensor image data throughput must be reduced by adjusting either the image frame rate, the resolution, the pixel depth, or a combination of these.

From the GC2145 sensor to the STM32MP13xx microprocessor through the STMIPID02 deserializer bridge, it is possible to continuously acquire images with the following resolutions and frames:

- 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

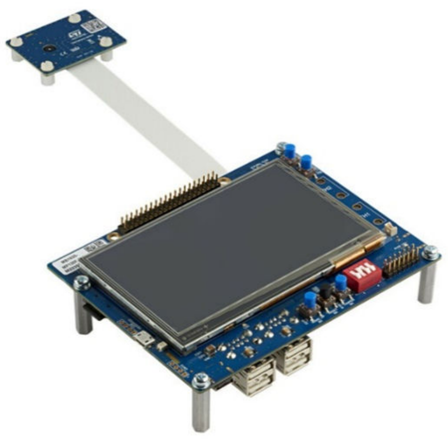
3.4 STMIPID02 Linux driver

The STMIPID02 MIPI CSI-2 deserializer bridge is designed to address a broad range of MIPI CSI-2 sensors targeting the consumer market and specifically mobile phone applications. To satisfy the growing demand to address this type of sensors from the industrial market to the IoT (Internet of Things) via artificial intelligence, the STMIPID02 driver has been upstreamed to the Linux community. It is freely available in Linux based applications. The STMIPID02 bridge driver is included in the STMicroelectronics OpenSTLinux delivery package, starting from version 1.1.0 and above.

4 Overall application

The camera demonstrator is part of the GTK demo launcher application, based on the OpenSTLinux software distribution package. It is ported on the STM32MP135F-DK Discovery board featuring the STM32MP135F and the STMIPID02, in link with the MB1897 camera board module featuring the GC2145 camera sensor thru the ribbon cable included in the STM32MP135F-DK board package.

Figure 2. **Camera demonstrator hardware**



4.1 STM32MP135F-DK Discovery board overview

The board main features:

- STM32MP135F 11x11 microprocessor
- STPMIC1D power module
- 2 × 512 Mbytes of DDR3L RAM
- STMIPID02 deserializer bridge
- STLINK-V3E through the USB Micro-B port (to connect a console to the host PC)
- 2× USB TypeC for flash load purpose and DRP via the STM32G0 MCU.
- Connectivity to receive the MB1897 GC2145 camera board module

For more information, refer to [R13].

4.2 MB1897 camera board module overview

The GC2145 camera sensor is soldered on the MB1897 circuit board and is connected through MIPI CSI-2 to the STM32MP135F for evaluation purpose.

4.3 The board image

The STMicroelectronics OpenSTLinux software targets STM32MP application boards (including the STM32MP135F-DK board) and provides drivers, library, tools, and examples to exercise STM32MP13x line embedded peripherals through its distribution package. It natively addresses MIPI CSI-2 camera sensors such as the GC2145 through the STMIPID02 MIPI CSI-2 deserializer bridge.

As for a demonstrator, the OpenSTLinux Starter package provides the board image file ready to be flashed in the STM32MP135F-DK board SDcard memory) and embeds the driver for either the GC2145 sensor or the OV5640 camera sensor (now deprecated) through the STMIPID02 deserializer bridge.

Two possible recommended image programming procedures are detailed in the next sections.

4.3.1 STM32CubeProgrammer software tool

This all-in-one software tool is an easy option to populate the binary image into any flash device connected to a STM32 MCU/MPU. The tool can be downloaded from [R10].

For a faster programming, it is recommended to position the board hardware boot switches to DFU (or USB boot mode) as indicated in the table below.

Table 2. **Boot switches configuration for programming**

Boot mode	Comments	Boot 2 (switch 3)	Boot 1 (switch 2)	Boot 0 (switch 1)
UART and USB	USB OTG	0	0	0

Once the binary image is programmed, position the hardware boot switches to SDCard boot mode.

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

It is recommended to program the whole binary image into an SDcard.

On a custom board, if the binary image partition must be split between the NOR-Flash (TF-A, U-boot) and *e*•MMC (Linux), refer to [R6].

4.3.2 Other programming methods

Here are two other programming methods:

- to program the binary image onto an SDcard using the conventional “dd” command, refer to the [R11] guidelines.
- to program the binary image onto a USB mass storage device using U-boot, refer to [R12].

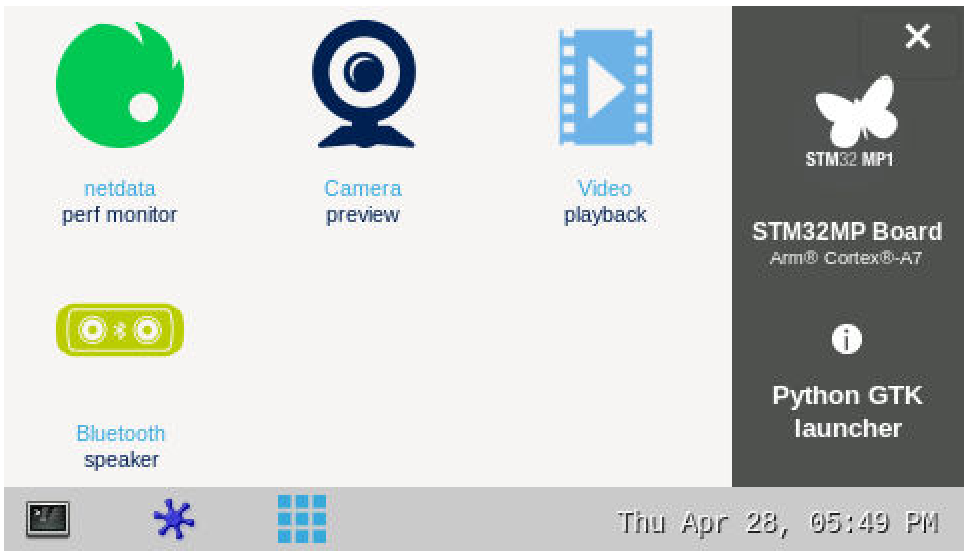
Note: *This method only applies if U-boot is available, therefore, if the two initial flash board partitions are already present into the SDCard or the board non-volatile memory.*

4.4 Booting the board image and activate camera preview display

Once the boot switches are changed to SDcard:

- The GTK launcher demonstrator screen should pop-up (see Figure 3).
- Select the camera preview logo to display the live video stream from the GC2145 MIPI CSI-2 camera module.

Figure 3. **GTK launcher demonstrator**



The script used to run the camera preview is in */usr/local/demo/application/camera/bin/launch_camera_preview.sh*. By default the sensor is configured in VGA RGB565 30fps, then the stream is scaled down to adapt to the board screen resolution (480 x 272).

The camera sensor format and framerate settings are adjusted through the V4l2 Linux kernel framework dedicated to the STM32MP13xx microprocessor and the GC2145 camera sensor. A subset of the commands is provided in the following section. For a comprehensive set of information on the camera subsystem topology and setup, refer to [R5].

4.5 V4L2 commands

On STM32MP13x, the camera subsystem must be configured first, before invoking any V4L2 application-related command. This is performed with the V4L2 `media-ctl` utility command (refer to [R6]).

Note: *The above step is a STM32MP13x specificity. On STM32MP15x products, V4l2 commands can directly interface with the camera sensor through I2C (refer to Section 2 [R15]).*

4.5.1 Configuring the camera sub-node and frame capture

The shell script below can be used to configure the GC2145 with one of the following format: RGB565_BE, RGB565_LE, YVYU, YUYV, UYVY, VYUY. BE/LE determine data endianes (only for RGB565 format).

The media-ctl part configures the camera subsystem while the the v4l2 command performs the video capture:

```
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
```

That is, to configure the sensor in VGA, the video format as RGB565_BE, and targeting 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: 81:2
looking up device: 81:0
looking up device: 81:3
looking up device: 81:4
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: 81:2
looking up device: 81:0
looking up device: 81:3
looking up device: 81:4
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: 81:2
looking up device: 81:0
looking up device: 81:3
looking up device: 81:4
Found 5 entities
Enumerating pads and links
Setting up format RGB565_2X8_LE 640x480 on pad dcmipp_parallel/1
Format set: RGB565_2X8_LE 640x480
Setting up format RGB565_2X8_LE 640x480 on pad dcmipp_dump_postproc/0
Format set: RGB565_2X8_LE 640x480
Opening media device /dev/media0
Enumerating entities
looking up device: 81:1
looking up device: 81:2
looking up device: 81:0
looking up device: 81:3
looking up device: 81:4
Found 5 entities
Enumerating pads and links
Setting up format RGB565_2X8_LE 640x480 on pad dcmipp_dump_postproc/1
Format set: RGB565_2X8_LE 640x480
Opening media device /dev/media0
Enumerating entities
looking up device: 81:1
looking up device: 81:2
looking up device: 81:0
looking up device: 81:3
looking up device: 81:4
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 Displaying a camera preview in Weston/Wayland

By default, the user is logged as root user. Since OpenSTLinux DV4.1, to launch Weston Wayland application and run weston/wayland commands, one need to be logged as weston user:

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

The camera subsystem must be set as described in 4.5.1 with the selected format, resolution and framerate. One can invoke for instance waylandsink in a gstreamer pipeline, using the same format as in the camera subsystem setup:

```
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...
```

The image preview from the GC2145 camera is displayed on the STM32MP13F-DK board LCD screen.

Revision history

Table 3. **Document revision history**

Date	Version	Changes
31-Jan-2023	1	Initial release.

Contents

List of tables

List of figures