PHYTEC Messtechnik GmbH

Application Note

How to use phyCAM camera modules with phyCORE-i.MX 8M Mini SBC on phyBOARD Polis

Revision History

Version	Changes	Author	Date
A0	Initial Release	H. Fendrich	02.03.2021

Content

1 Overview

The i.MX 8M Mini Microcontroller supported one MIPI CSI-2 camera interfaces (see figure 1).

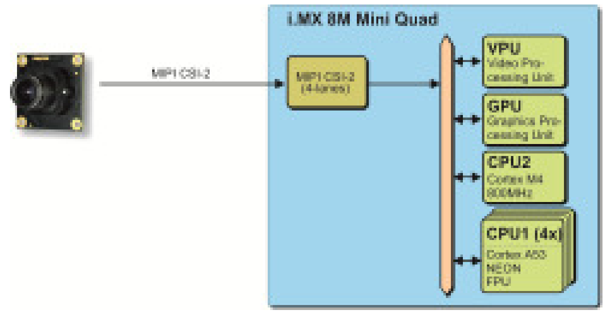
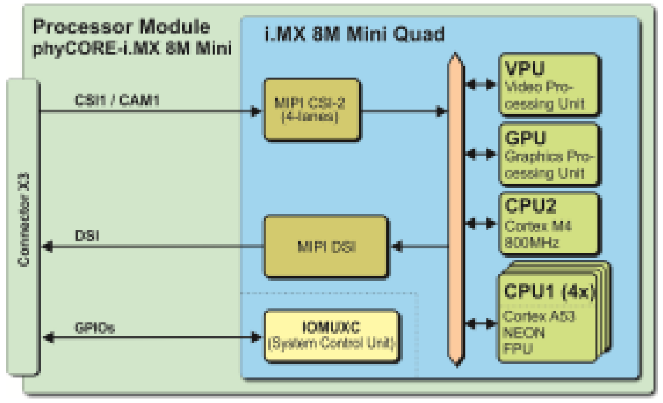

i.MX 8M Mini Quad
MIPI CSI-2
MIPI CSI-2 (4-lanes)
VPU Video Processing Unit
GPU Graphics Processing Unit
CPU2 Cortex M4 800MHz
CPU1 (4x) Cortex A53 NEON FPU

figure 1: Block Diagram Camera Interfaces of i.MX 8M Mini Controller (Quad)

On the phyCORE-i.MX 8M Mini the CSI1/CAM1 camera path go out as CSI-2 MIPI signal. (see **Fehler! Verweisquelle konnte nicht gefunden werden.**)


Processor Module phyCORE-i.MX 8M Mini
i.MX 8M Mini Quad
Connector X3
CSI1 / CAM1
DSI
GPIOs
MIPI CSI-2 (4-lanes)
MIPI DSI
IOMUXC (System Control Unit)
VPU Video Processing Unit
GPU Graphics Processing Unit
CPU2 Cortex M4 800MHz
CPU1 (4x) Cortex A53 NEON FPU

On the PHYTEC or customer carrier boards can the interfaces are led out as phyCAM-M. For more information to phyCAM-M see the phyCAM-Manuals on

PHYTEC Homepage.
On the phyBOARD-Polis baseboard is the camera interfaces led out as phyCAM-M (MIPI CSI-2) interfaces (see figure 2). Here you can connect different phyCAM-M camera modules.

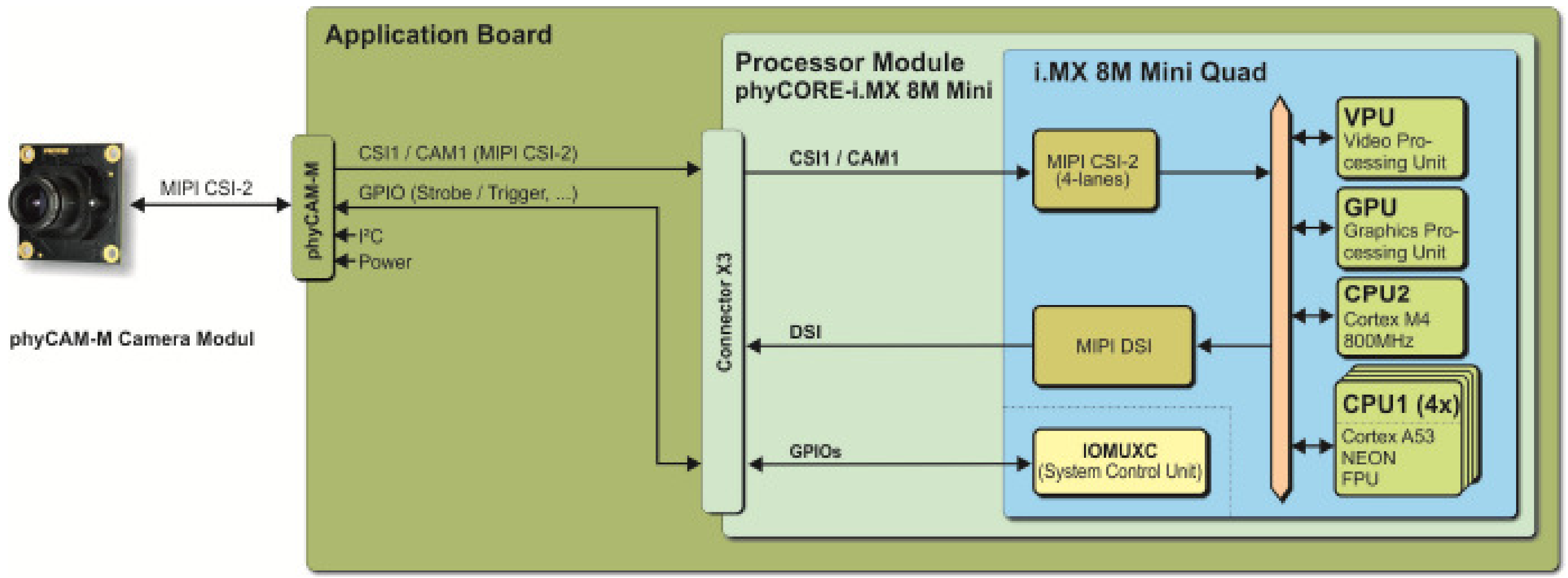


figure 2: Block Diagram of phyCAM-M Camera Interfaces of phyCORE-i.MX 8M Mini (Quad) and the go out on the phyBOARD-Polis-i.MX 8M Mini - SBC

The BSP shipped with the Kit includes already the software drivers for the supported phyCAM-M camera modules. The drivers are compatible with v4l2. Also GStreamer scripts are included for the evaluation of the camera modules. If you need the camera interface to connecting your own camera module, is an adapter to phyCAM-M necessary.

2 Camera Connectors on the Polis - Carrier Boards

The development kits for the phyBOARD-Polis-i.MX 8M Mini contain:

- one carrier board (Polaris)
- one phyCORE-i.MX 8M modul SOM

The phyCORE-i.MX 8M Mini is direct soldering on carrier board.

On the base board Polis (PB-02820) we convert the MIPI CSI-2 interfaces in phyCAM-M standard.

- phyCAM-M camera interface 1 (support on connector X10)

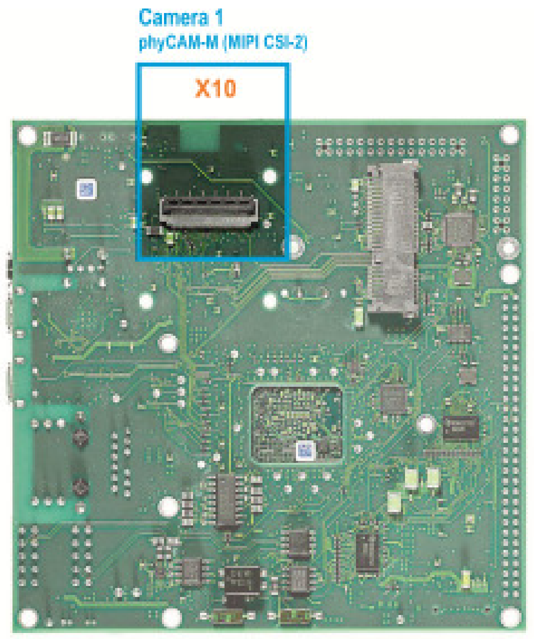


figure 3: Camera Interfaces on phyBOARD-Polis up PCB Version PL1532.1 for the phyCORE-i.MX 8M Mini SOM

Note: If you use the phyCAM-M interface, use an FFC cable that is especially suitable for FH41 connectors (e.g. Phytec WF271). Standard FFC cables can cause a short circuit.

3 Supported camera boards (YOCTO Linux PD21.1.0 and higher)

The cameras and the camera interface supported as a sub-devices. To configure the v4l2 framework is the handler "media-ctl" used. See chapter 5

The device tree of the camera VM-016 (sensor AR0144) is preselected in PD21.1.0. To use other cameras (e.g. VM-017/117) the device tree must be changed. See chapter 0

3.1 Supported camera types in standard vision image (phytec-vision-image-phyboard-polis-imx8mm-3)

Per default are following Kameratyps are supported in the image:
- VM-016-BW-M series (VM-016-BW-M, VM-016-BW-M-M12, VM-016-BW-M-H, …) based on camera sensor “AR0144”
- VM-016-COL-M series (VM-016-COL-M, VM-016-COL-M-M12, VM-016-COL-M-H, …) based on camera sensor “AR0144”

After login, you can start working with the demo-scripts. (see chapter 4.1)

Hardware Configuration		**Possible Parameters**				**default I²C-address (default jumper on camera and boards)**
phyCAM camera model (part number)	**connected to**	**csi_port**	**cam_bus_type**	**cam_type**	**cam_i2c_address**	
VM-016-BW-M (-M12 / -H)	X10 on Polaris board	1	phyCAM-M	VM-016	0x10, 0x18	0x10
VM-016-COL-M (-M12 / -H)	X10 on Polaris board	1	phyCAM-M	VM-016	0x10, 0x18	0x10

Notes:
- I²C addresses of the camera are set by hardware configuration (jumper setting on the camera and / or on the baseboard. Please refer to the VM-016-M manual L-1018 and the hardware manual of the Polis board L-862.

3.2 Change the camera types in standard vision image (phytec-vision-image-phyboard-polis-imx8mm-3)

The following camera types are interchangeable in the image:

- VM-017-BW-M series (VM-017-BW-M, VM-017-BW-M-M12, VM-017-BW-M-H, …) based on camera sensor “AR052x”
- VM-017-COL-M series (VM-017-COL-M, VM-017-COL-M-M12, VM-017-COL-M-H, …) based on camera sensor “AR052x”
- VM-117-BW-M series (VM-117-BW-M, VM-117-BW-M-M12, …) based on camera sensor “AR052x”
- VM-117-COL-M series (VM-117-COL-M, VM-117-COL-M-M12, …) based on camera sensor “AR052x”
- VM-017-BW-L series (VM-017-BW-L, VM-017-BW-L-M12, VM-017-BW-L-H, …) based on camera sensor “AR052x” (note: in progress)
- VM-017-COL-L series (VM-017-COL-L, VM-017-COL-L-M12, VM-017-COL-L-H, …) based on camera sensor “AR052x” (note: in progress)

Hardware Configuration		**Possible Parameters**				**default I²C-address (default jumper on camera and boards)**
phyCAM camera model (part number)	**connected to**	**csi_port**	**cam_bus_type**	**cam_type**	**cam_i2c_address**	
VM-017-BW-M (-M12 / -H) VM-117-BW-M (-M12)	X10 on Polaris board	1	phyCAM-M	VM-017	0x36, 0x37	0x36
VM-017-COL-M (-M12 / -H) VM-117-COL-M (-M12)	X10 on Polaris board	1	phyCAM-M	VM-017	0x36, 0x37	0x36
VM-017-BW-L (-M12 / -H)	X10 on Polaris board via VZ-018	1	phyCAM-M	VM-017	0x36, 0x37	0x36
VM-017-COL-L (-M12 / -H)	X10 on Polaris board via VZ-018	1	phyCAM-M	VM-017	0x36, 0x37	0x36

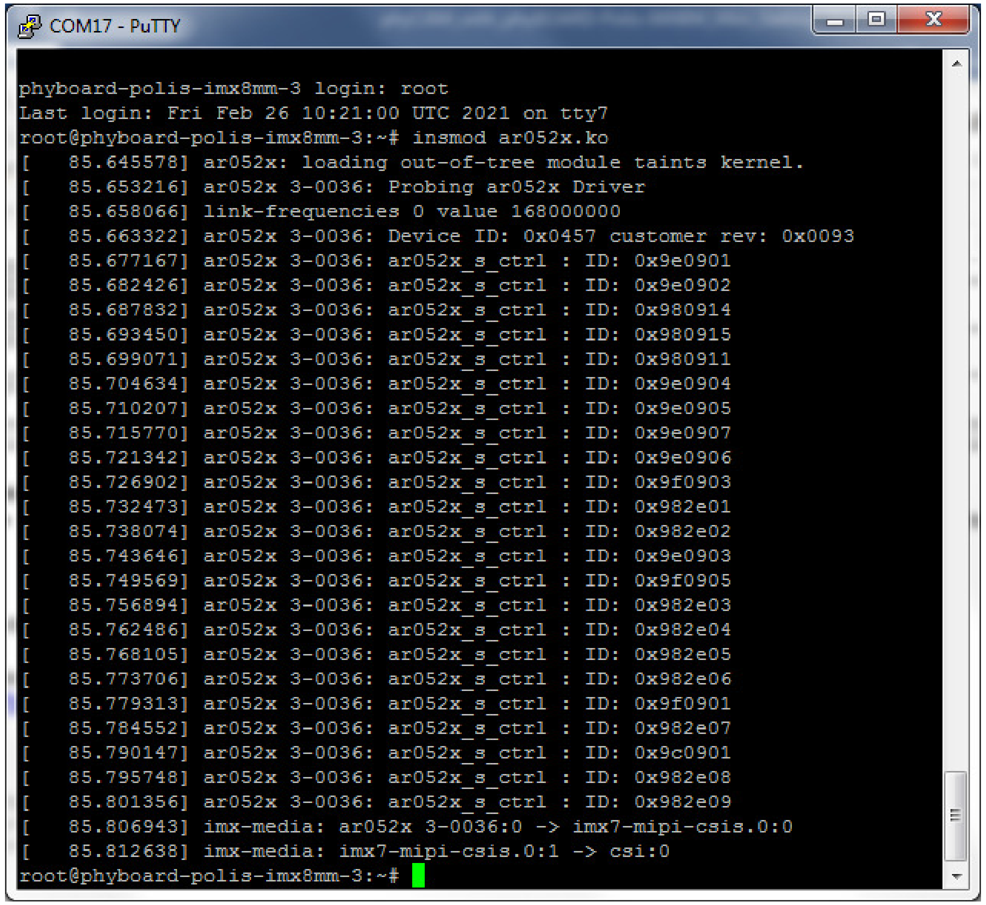
Notes:

- I²C addresses of the camera are set by hardware configuration (jumper setting on the camera and / or on the baseboard. Please refer to the VM-x17-_ manuals L-1020/1021/1022 and the hardware manual of the Polis board L-862.

How to change the device tree for VM-017/VM-117

Please following the instructions of “README.md” from PHYTEC FTP Server
ftp://ftp.phytec.de/pub/ImageProcessing/phyBOARD-Polis-i.MX8MM_linux_PD21.1.0/VM-017/

After reboot and login, load ar052x.ko camera driver: > insmod ar052x.ko



```
phyboard-polis-imx8mm-3 login: root
Last login: Fri Feb 26 10:21:00 UTC 2021 on tty7
root@phyboard-polis-imx8mm-3:~# insmod ar052x.ko
[   85.645578] ar052x: loading out-of-tree module taints kernel.
[   85.653216] ar052x 3-0036: Probing ar052x Driver
[   85.658066] link-frequencies 0 value 168000000
[   85.663322] ar052x 3-0036: Device ID: 0x0457 customer rev: 0x0093
[   85.677167] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9e0901
[   85.682426] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9e0902
[   85.687832] ar052x 3-0036: ar052x_s_ctrl : ID: 0x980914
[   85.693450] ar052x 3-0036: ar052x_s_ctrl : ID: 0x980915
[   85.699071] ar052x 3-0036: ar052x_s_ctrl : ID: 0x980911
[   85.704634] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9e0904
[   85.710207] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9e0905
[   85.715770] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9e0907
[   85.721342] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9e0906
[   85.726902] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9f0903
[   85.732473] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e01
[   85.738074] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e02
[   85.743646] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9e0903
[   85.749569] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9f0905
[   85.756894] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e03
[   85.762486] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e04
[   85.768105] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e05
[   85.773706] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e06
[   85.779313] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9f0901
[   85.784552] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e07
[   85.790147] ar052x 3-0036: ar052x_s_ctrl : ID: 0x9c0901
[   85.795748] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e08
[   85.801356] ar052x 3-0036: ar052x_s_ctrl : ID: 0x982e09
[   85.806943] imx-media: ar052x 3-0036:0 -> imx7-mipi-csis.0:0
[   85.812638] imx-media: imx7-mipi-csis.0:1 -> csi:0
root@phyboard-polis-imx8mm-3:~#
```

figure 4: output after load VM-017/117 driver AR052x

Now the VM-0x17 driver (AR052x) is ready for use.

Now you can start working with the demo-scripts. (see chapter 4.1)

4 Demo scripts

If you want to see the live images, we recommend using a display on the Polis Board (for example 10” 1280x800 Display KPEB-AV-10-100.A0)

There are 3 sub directories with demo scripts for the cameras:

- gstreamer-examples
- v4l2_c-examples
- opencv-examples

Notes: Remove the qt-demo at the first start. With the \gstreamer-examples\remove_qt_demo.sh script, else the qt-demo is alway present on the display.

4.1 GStreamer scripts

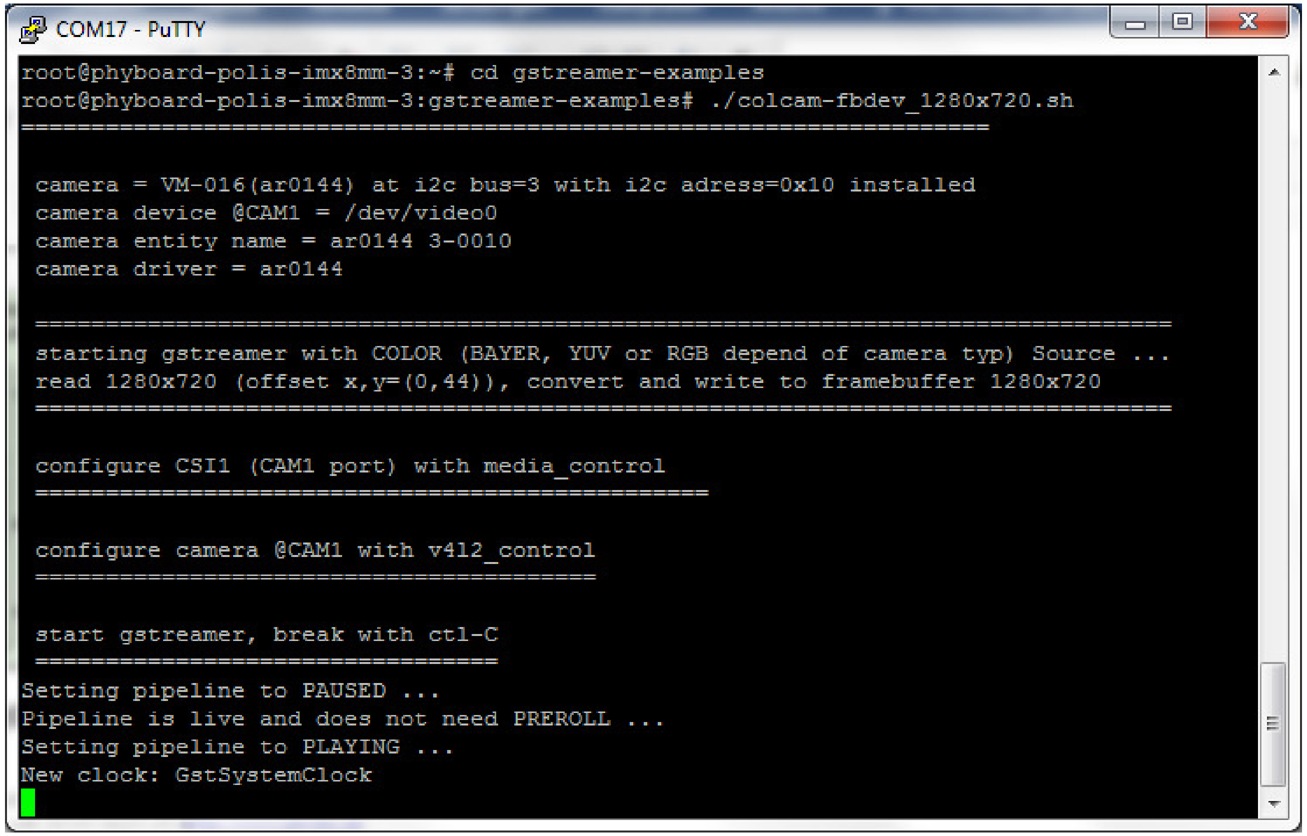
After login, change into the directory: \gstreamer-examples\..
cd gstreamer_examples <ENTER>.

Now you can start working with the GStreamer demo-scripts.
Start the scripts with den word phrase "col" or "bw" depending on the connected camera color type.

- _cam-fbdev_1280x720.sh – scripts, show a livestream on display
- _cam-save_jpg_full_res.sh – scripts, save a JPG File in this directory
- _cam-save_raw_full_res.sh – scripts, save a RAW File in this directory
- func.sh – script, detect the camera typ and define the parameter for the scripts
- remove_qt_demo.sh – script, remove the qt-demo from autostart
- turn_off_on_wayland.sh – script, turn of/on if wayland necessary or not

Subdirectories:

- more_ar0144_scripts: contain more scripts for the VM-016 camera series (v4l-ctrl_ar0144.txt, list the v4l2-controls for this camera)
- more_ar052x_scripts: contain more scripts for the VM-017/117 camera series (v4l-ctrl_ar052x.txt, list the v4l2-controls for this camera)
- vpu_enc_dec_scripts: contain scripts how use the vpu-encoder (e.g networkstreaming or save H.264 streams)
- tools: contain scripts to get and set the camera register direct via i2c access
- phytec_usb_cam: scripts for use the Phytec USB-cameras



```
root@phyboard-polis-imx8mm-3:~# cd gstreamer-examples
root@phyboard-polis-imx8mm-3:gstreamer-examples# ./colcam-fbdev_1280x720.sh
=====================================================================

 camera = VM-016(ar0144) at i2c bus=3 with i2c adress=0x10 installed
 camera device @CAM1 = /dev/video0
 camera entity name = ar0144 3-0010
 camera driver = ar0144

 ====================================================================================
 starting gstreamer with COLOR (BAYER, YUV or RGB depend of camera typ) Source ...
 read 1280x720 (offset x,y=(0,44)), convert and write to framebuffer 1280x720
 ====================================================================================

 configure CSI1 (CAM1 port) with media_control
 ================================================

 configure camera @CAM1 with v4l2_control
 ========================================

 start gstreamer, break with ctl-C
 =================================
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
```

figure 5: call of “colcam-fbdev_1280x720.sh” for VM-016-COL (AR0144) series (A live image should show on the display)

All camera/video components get a separate "/dev/video[x]" or "dev/v4l-subdev[x]" device.
The v4l2 - capabilities are showed if you type: "v4l2-ctl -d [device] -L" e.g. "v4l2-ctl -d /dev/video0 -L".

For the first use, the camera and controller-camera interface must be configured with the tool v4l2-ctl. See chapter 5.

4.2 Scripts to call C/C++ files based on v4l2 interface

After login, change into the directory: \v4l2_c-examples\..
cd v4l2_c-examples <ENTER>.

Now you can start working with the demo-scripts.
Start the scripts with den word phrase "col" or "bw" depending on the connected camera color type.

- ar0144_col/bw_full_save-raw – scripts, save a raw image from VM-016 in full resolution (8/10 and 12-Bit Formats are possible)
- ar052x_col/bw_full_save-raw – scripts, save a raw image from VM-017/117 in full resolution (8/10 and 12-Bit Formats are possible)

For saving the image we use the program Yavta. Yavta stands for "Yet Another V4L2 Test Application". This is a test application based on V4L2 Linux interface.

A other way to use in your C-program is the direct access call to the v4l2 interface. For example:

- *v4l2-ctl -d0 --stream-mmap --stream-count 1 --stream-to=raw_image1.raw*

For the first use, the camera and controller-camera interface must be configured with the tool v4l2-ctl. See chapter 5.

4.3 OpenCV scripts

PHYTEC i.MX 8M Mini BSP PD21.1.0 includes OpenCV4.4. For use OpenCV with image output we use window manager Wayland. PHYTEC tested openCV with PYTHON programming language. Examples of scripts can be found in the path “opencv-examples”:

- /opencv-examples/python/python3 phycam_video_v4l2.py: show the live image from the camera
- /opencv-examples/python/python3 face_detection.py: small application to detect faces

For the first use, the camera and controller-camera interface must be configured with the tool v4l2-ctl. See chapter 5.

5 Configure camera and controller-camera-interface with “media-ctl”

Modern System-on-Chip (SoC) devices support a wide range of functionality in the way of internal hardware blocks which has resulted in a variety of ways to interconnect functions within the SoC that alter the V4L device content.

The Media Controller kernel API has been designed to expose detailed information about media devices and capture devices to userspace in a way that allows them to be interconnected in dynamic and complex ways at runtime.

Media controller devices expose entities to the media controller framework. Each entity has one or more source pads and one or more sink pads. You use the media controller kernel API (ie via media-ctl) to query entities, link source pads to sink pads, and set formats of pads.

We use media-ctl to set:

- the desired resolution and color format of the camera sensor
- the fitting resolution and color format in the CSI interface (controller-camera-interface)
- the way (and/or preprocessing) through the different hardware blocks in the controller
- the resulting camera stream is made available as “/dev/videox” device

The settings are necessary:

- once after restarting the system and using the camera for the first time
- set a new resolution, color format or bit-depth
- set a new path through the hardware blocks

Following entities are present at the i.MX 8M Mini Image (check with “media-ctl –p”):

- entity 1: csi (2 pads, 2 links)
  pad0: Sink
  pad1: Source
- entity 4: csi capture (1 pad, 1 link)
  pad0: Sink
- entity 10: imx7-mipi-csis.0 (2 pads, 2 links)
  pad0: Sink
  pad1: Source
- entity 13: ar0144 3-0010 (1 pad, 1 link)   #(depend on selected camera)
  pad0: Source

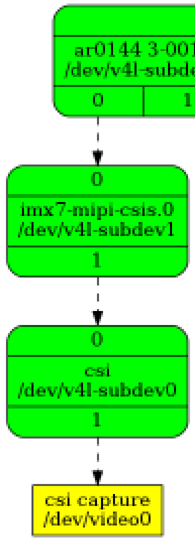
For setting the media-ctl we use in our demo script following points:

- media-ctl –r	reset all links to inactive
- media-ctl –l	setup the links between the hardware blocks
- media-ctl –V	formats the hardware blocks

For example we will set the VM-016-COL (AR0144) with full Resolution from CSI Port to a /dev/videox device:

First we set the links:

```
media-ctl -r
media-ctl -l "'ar0144 3-0010':0->'imx7-mipi-csis.0':0[1]"
media-ctl -l "'imx7-mipi-csis.0':1->'csi':0[1]"
media-ctl -l "'csi':1->'csi capture':0[1]"
#         Camera -> imx7_mipi_csis.0 -> CSI -> CSI capture (/dev/videoX)
```



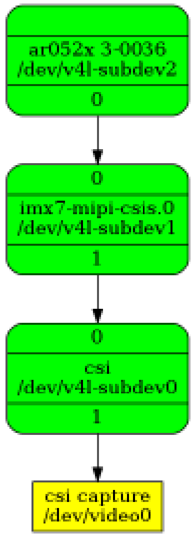
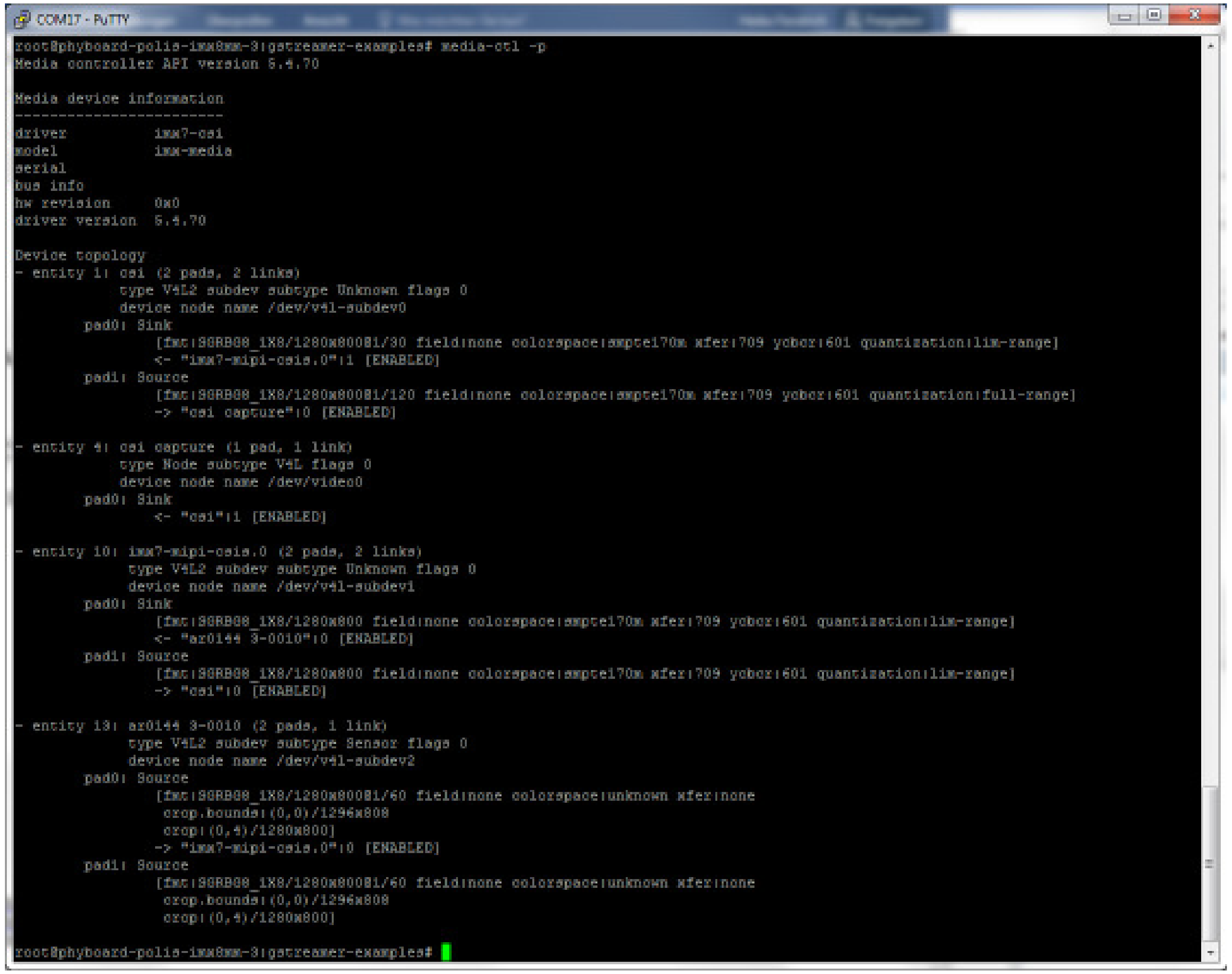


figure 6: video path VM-016 (AR0144)   video path VM-017 (AR052x)

Then we formats the hardware blocks:

```
media-ctl -V "'ar0144 3-0010':0 [fmt:SGRBG8_1X8/1280x800 (0,4)/1280x800]"
media-ctl -V "'imx7-mipi-csis.0':1 [fmt:SGRBG8_1X8/1280x800]"
media-ctl -V "'csi':1 [fmt:SGRBG8_1X8/1280x800 field:none]"
```

Check the settings with “media-ctl –p”:



```
root@phyboard-polis-imx8mm-3:gstreamer-examples# media-ctl -p
Media controller API version 5.4.70

Media device information
------------------------
driver          imx7-csi
model           imx-media
serial
bus info
hw revision     0x0
driver version  5.4.70

Device topology
- entity 1: csi (2 pads, 2 links)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev0
        pad0: Sink
                [fmt:SGRBG8_1X8/1280x800@1/30 field:none colorspace:smpte170m xfer:709 ycbcr:601 quantization:lim-range]
                <- "imx7-mipi-csis.0":1 [ENABLED]
        pad1: Source
                [fmt:SGRBG8_1X8/1280x800@1/120 field:none colorspace:smpte170m xfer:709 ycbcr:601 quantization:full-range]
                -> "csi capture":0 [ENABLED]

- entity 4: csi capture (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video0
        pad0: Sink
                <- "csi":1 [ENABLED]

- entity 10: imx7-mipi-csis.0 (2 pads, 2 links)
             type V4L2 subdev subtype Unknown flags 0
             device node name /dev/v4l-subdev1
        pad0: Sink
                [fmt:SGRBG8_1X8/1280x800 field:none colorspace:smpte170m xfer:709 ycbcr:601 quantization:lim-range]
                <- "ar0144 3-0010":0 [ENABLED]
        pad1: Source
                [fmt:SGRBG8_1X8/1280x800 field:none colorspace:smpte170m xfer:709 ycbcr:601 quantization:lim-range]
                -> "csi":0 [ENABLED]

- entity 13: ar0144 3-0010 (2 pads, 1 link)
             type V4L2 subdev subtype Sensor flags 0
             device node name /dev/v4l-subdev2
        pad0: Source
                [fmt:SGRBG8_1X8/1280x800@1/60 field:none colorspace:unknown xfer:none
                 crop.bounds:(0,0)/1296x808
                 crop:(0,4)/1280x800]
                -> "imx7-mipi-csis.0":0 [ENABLED]
        pad1: Source
                [fmt:SGRBG8_1X8/1280x800@1/60 field:none colorspace:unknown xfer:none
                 crop.bounds:(0,0)/1296x808
                 crop:(0,4)/1280x800]

root@phyboard-polis-imx8mm-3:gstreamer-examples#
```

figure 7: Output “media-ctl –p”:

Now the Camera is present at “dev/video0” device.
The camera is ready for access.

6 De-Bayering (demosaicking) with NEON CoProcessor

Most of CMOS color chips provide the image in the bayer mosaicing (bayer raw) format. For get a color image in RGB format is it necessary to convert the bayer raw image.
- https://en.wikipedia.org/wiki/Bayer_filter
- https://de.wikipedia.org/wiki/Bayer-Sensor

There are exist different algorithm for converting.
- https://en.wikipedia.org/wiki/Demosaicing

If the microprocessor does not include debayering hardware, have to do the converting via software. For this you need additional processing power and the framerate goes down. It is better to use the NEON coprocessor of the i.MX 8M Mini. For this support PHYTEC a special function.
It is present as GStreamer plugin "bayer2rgbneon" and in sources for use in an own C-program. We support a simple bilinear algorithm.

For use in GStreamer take "bayer2rgbneon" plugin. For more information to "bayer2rgbneon" parameters type:
- gst-inspect-1.0 bayer2rgbneon

The source are located at:
- https://git.phytec.de/bayer2rgb-neon/