ST Restriced – Subject to Non-Disclosure Agreement – Do not copy

V4L2 camera overview

ST Restriced – Subject to Non-Disclosure Agreement – Do not copy

Stable: 26.02.2021 - 17:17 / Revision: 26.02.2021 - 17:18

A quality version of this page, approved on *26 February 2021*, was based off this revision.

This article gives information about the Linux®V4L2 camera framework.

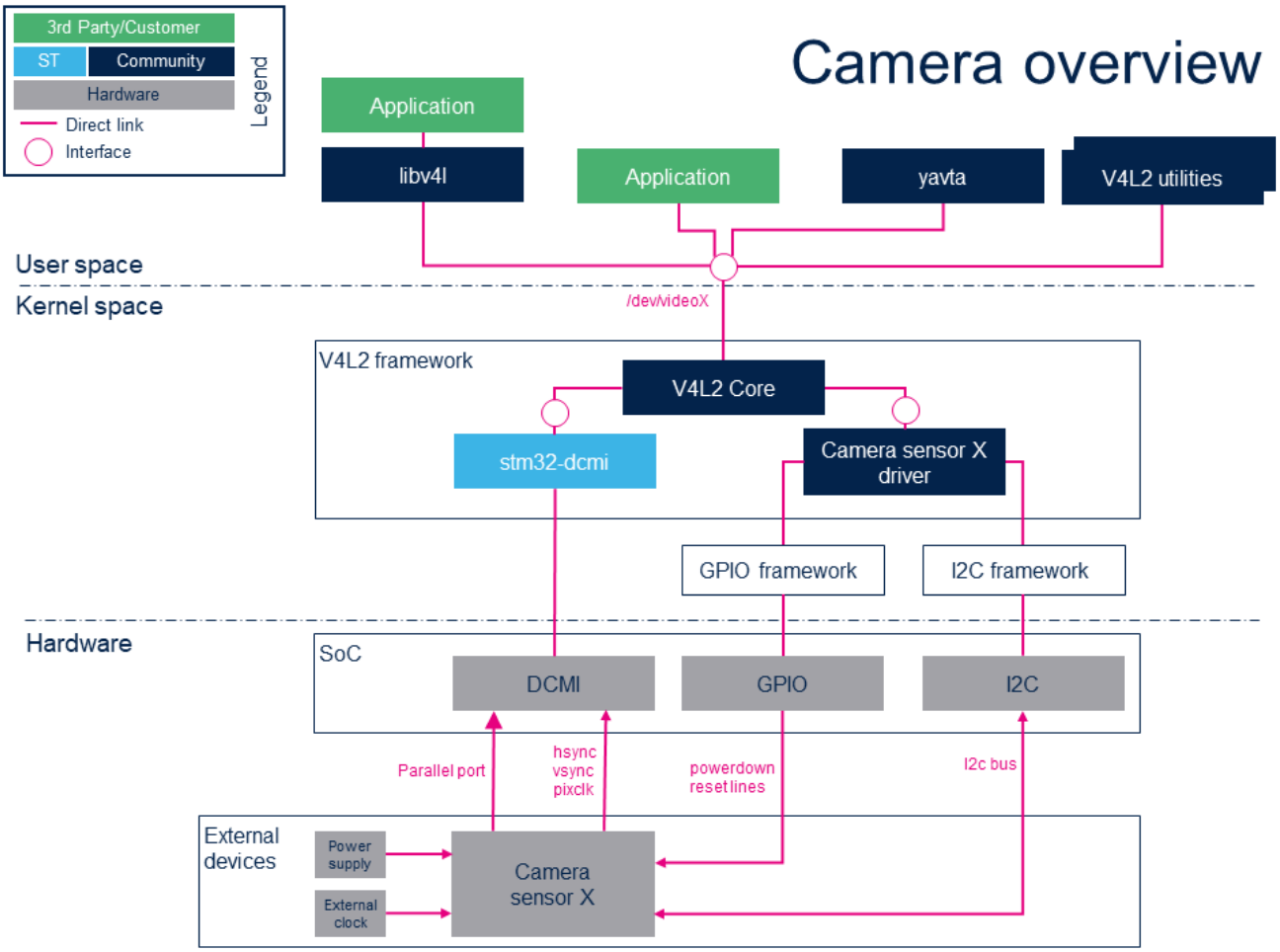
Contents

1 Framework purpose

The V4L2 Linux kernel framework[1] allows to control both an external camera sensor and the camera interface in order to capture raw frames in various pixel formats or encoded stream data such as JPEG.

This could be typically used, with the help of other Linux multimedia frameworks and applications, to take snapshot, to make preview, to make a video recording or even remotely stream images from the camera sensor.

2 System overview



2.1 Component description

- **Application** (User space)

Any application relying on V4L2 Linux kernel interface or libv4l abstraction layer. GStreamer framework provides such application.

- **V4L2 utilities** (User space)

A set of tools to test, configure and use the whole camera subsystem, including the external camera sensor and the camera interface. V4l2-ctl is one of the most usefull utility.

- **V4L2 libraries (libv4l)** (User space)

A set of libraries on top of the V4L2 Linux kernel interface which abstract the kernel interface in order to simplify, keep compatibility or add some hooks between V4L-based applications and the V4L2 kernel interface.

- **yavta** (User space)

Yavta is a test tool which relies on the V4L2 Linux kernel interface.

- **V4L2 core** (Kernel space)

This layer represents the standard Linux kernel V4L2 Framework.

- **stm32-dcmi** (Kernel space)

This V4L2 DCMI Linux device driver handles the DCMI hardware block.

- **Camera sensor X driver** (Kernel space)

This V4L2 Linux device driver handles the camera sensor X external peripheral, handles some GPIOs lines and potentially power supplies to power-up/down the camera sensor. The communication with camera sensor is done through the i2c bus.

- **DCMI** (Hardware)

The Digital Camera Memory Interface hardware block.

- **Camera sensor X** (Hardware)

The camera sensor external peripheral.

2.2 APIs description

The V4L2 userland API is documented in the Linux Media subsystem documentation[2]

The V4L2 kernel framework internal API is documented in the V4L2 Kernel Support section of the Linux Kernel documentation[3]

ST Restriced – Subject to Non-Disclosure Agreement – Do not copy

3 Configuration

3.1 Kernel configuration

The STM32 camera interface and OV5640 camera sensor are enabled by default in STMicroelectronics deliveries.

Nevertheless this is not the case when using upstream kernel version. In this case, the DCMI V4L2 driver can be enabled using Linux kernel menuconfig tool:

```
[*] Device Drivers --->
    [*] Multimedia support --->
        [*] V4L platform devices  --->
            [*] STM32 Digital Camera Memory Interface (DCMI) support
```

The external camera sensor connected to the camera interface must also be enabled, here is an example with the OV5640 Omnivision camera sensor located on the MB1379 camera daughter board[4] connected to the CN7 camera connector[5] of the STM32MP15 evaluation board[6]:

```
[*] Device Drivers --->
    [*] Multimedia support --->
        I2C Encoders, decoders, sensors and other helper chips  --->
            [*] OmniVision OV5640 sensor support
```

3.2 Device tree configuration

Refer to DCMI device tree configuration article for a complete view of DCMI & sensor configuration thanks to Linux kernel device tree mechanism.

4 How to use the framework

The use cases described here are enabled using V4l2-ctl, gst-launch or gst-play command line utilities.

4.1 List the video devices and their capabilities

List all the available video devices using **--list-devices** option:

```
Board $> v4l2-ctl --list-devices
```

```
STM32 Camera Memory Interface (platform:dcmi):
        /dev/video0
```

If several devices are available, use **-d** option after any V4l2-ctl commands to target a specific device. If **-d** option is not specified, /dev/video0 is targeted by default.

In order to have information on a specific device, use **-D** option:

```
Board $> v4l2-ctl -d /dev/video0 -D
```

```
Driver Info (not using libv4l2):
        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
```

4.2 Controlling camera

Use V4l2-ctl with **-L** option to get the list of supported controls:

```
Board $> v4l2-ctl -L
```

```
User Controls

                     contrast (int)    : min=0 max=255 step=1 default=0 value=0
flags=slider
                   saturation (int)    : min=0 max=255 step=1 default=64 value=64
flags=slider
                          hue (int)    : min=0 max=359 step=1 default=0 value=0
flags=slider
        white_balance_automatic (bool)   : default=1 value=1 flags=update
                   red_balance (int)    : min=0 max=4095 step=1 default=0 value=128
flags=inactive, slider
                  blue_balance (int)    : min=0 max=4095 step=1 default=0 value=128
flags=inactive, slider
                     exposure (int)    : min=0 max=65535 step=1 default=0 value=885
flags=inactive, volatile
                gain_automatic (bool)   : default=1 value=1 flags=update
                          gain (int)    : min=0 max=1023 step=1 default=0 value=32
flags=inactive, volatile
               horizontal_flip (bool)   : default=0 value=0
                 vertical_flip (bool)   : default=0 value=0

Camera Controls

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

Image Processing Controls

                  test_pattern (menu)   : min=0 max=1 default=0 value=0
                               0: Disabled
                               1: Color bars
```

Information

"value=" field returns the current value of the control

The control value can be changed thanks to **--set-ctrl** option, for example:

```
Board $> v4l2-ctl --set-ctrl test_pattern=1
```

The control value can be changed dynamically. In the following example, the color bar is enabled/disabled while preview is running:

- Start preview in background

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=1280, Height=720, framerate=
(fraction)15/1" ! queue ! autovideosink -e &
```

- Then alternate the color bar activation or not

```
Board $> v4l2-ctl --set-ctrl test_pattern=1;sleep 1;v4l2-ctl --set-ctrl test_pattern=0;
sleep 1;v4l2-ctl --set-ctrl test_pattern=1;sleep 1;v4l2-ctl --set-ctrl test_pattern=0;
killall gst-launch-1.0
```

- Horizontal/vertical flip can also be changed while preview is running:

```
Board $> v4l2-ctl --set-ctrl horizontal_flip=1;sleep 2;v4l2-ctl --set-ctrl
horizontal_flip=0;sleep 2;v4l2-ctl --set-ctrl vertical_flip=1;sleep 2;v4l2-ctl --set-ctrl
vertical_flip=0;killall gst-launch-1.0
```

4.3 Set the pixel format, resolution and framerate

Use --list-formats-ext option to get the supported pixel format, resolution and framerate:

```
Board $> v4l2-ctl --list-formats-ext
```

```
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'JPEG' (compressed)
        Name        : JFIF JPEG
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.067s (15.000 fps)

        Index       : 1
        Type        : Video Capture
        Pixel Format: 'UYVY'
        Name        : UYVY 4:2:2
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
```

```
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.067s (15.000 fps)

        Index       : 2
        Type        : Video Capture
        Pixel Format: 'YUYV'
        Name        : YUYV 4:2:2
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.067s (15.000 fps)

        Index       : 3
        Type        : Video Capture
        Pixel Format: 'RGBP'
        Name        : 16-bit RGB 5-6-5
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
```

```
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.067s (15.000 fps)
```

In order to change the camera configuration, select first the framerate using **--set-parm** option:

```
Board $> v4l2-ctl --set-parm=30
```

Then select the wanted resolution and/or pixel format using **--set-fmt-video** option:

```
Board $> v4l2-ctl --set-fmt-video=width=320,height=240,pixelformat=RGBP
```

4.4 Set the framerate

- With V4l2-ctl, use **--set-parm** option giving the framerate numerator only, the denominator is fixed to 1 (only integer framerate values are allowed):

```
Board $> v4l2-ctl --set-parm=<framerate numerator>
```

Take 100 VGA pictures at 30fps:

```
Board $> v4l2-ctl --set-parm=30;v4l2-ctl --set-fmt-video=width=640,height=480,
pixelformat=JPEG --stream-mmap --stream-count=100 --stream-to=pics@30fps.jpeg
```

Replay at 30fps using gst-play:

```
Board $> gst-play-1.0 pics@30fps.jpeg --videosink="videorate ! video/x-raw, framerate=
(fraction)30/1 ! autovideosink"
```

Take 100 VGA pictures at 15fps:

```
Board $> v4l2-ctl --set-parm=15;v4l2-ctl --set-fmt-video=width=640,height=480,
pixelformat=JPEG --stream-mmap --stream-count=100 --stream-to=pics@15fps.jpeg
```

Replay at 15fps using gst-play:

```
Board $> gst-play-1.0 pics@15fps.jpeg --videosink="videorate ! video/x-raw, framerate=
(fraction)15/1 ! autovideosink"
```

- With GStreamer, using **framerate** caps:

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, ... framerate=(fraction)<numerator>
/<denominator>" ! ...
```

Preview VGA@30fps

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=640, Height=480, framerate=
(fraction)30/1" ! queue ! autovideosink -e
```

Preview VGA@15fps

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=640, Height=480, framerate=
(fraction)15/1" ! queue ! autovideosink -e
```

4.5 Grab a raw frame

Capture a QVGA RGB565 raw frame on disk:

```
Board $> v4l2-ctl --set-fmt-video=width=320,height=240,pixelformat=RGBP --stream-mmap --
stream-count=1 --stream-to=grab-320x240-rgb565.raw
```

In order to display it, this raw frame must be converted first to JPEG:

```
Board $> gst-launch-1.0 filesrc location= grab-320x240-rgb565.raw blocksize=153600 !
"video/x-raw, format=(string)RGB16, width=(int)320, height=(int)240, framerate=(fraction)
30/1" ! videoconvert ! jpegenc ! filesink location=grab-320x240-rgb565.jpeg
```

Then weston-image utility can be used to display this JPEG file:

```
Board $> weston-image grab-320x240-rgb565.jpeg
```

4.6 Fullscreen preview

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=1280, Height=720, framerate=
(fraction)15/1" ! queue ! autovideosink -e
```

Information

Please note that GStreamer overwrites all the parameters that could have been previously set on the video device (for ex. parameters set through V4l2-ctl commands such as resolution, pixel format, framerate, ...)

4.7 Take a picture

Capture a 5Mp JPEG:

```
Board $> v4l2-ctl --set-parm=15; v4l2-ctl --set-fmt-video=width=2592,height=1944,
pixelformat=JPEG --stream-mmap --stream-count=1 --stream-skip=3 --stream-to=pic-5Mp.jpeg;
v4l2-ctl --set-parm=30
```

Then display it:

```
Board $> weston-image pic-5Mp.jpeg
```

You can check the picture resolution using gst-typefind:

```
Board $> gst-typefind-1.0 pic-5Mp.jpeg
```

```
pic-5Mp.jpeg - image/jpeg, width=(int)2592, height=(int)1944, sof-marker=(int)0
```

4.8 Pictures streaming over network

Refer to How to stream camera over network article to get some examples on how to stream camera content over network.

5 How to trace and debug

5.1 How to monitor

5.1.1 Check of devicetree configuration

Here are some commands to verify that DCMI is enabled, check which camera sensor is used and log other details about devicetree settings:

```
 rm devicetree.txt
 echo "[devicetree]" >> devicetree.txt
 echo "|-[dcmi]" >> devicetree.txt
 find /proc/device-tree/soc | grep 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
 find /proc/device-tree/soc | grep camera | sed 's/\/proc\/device-tree\/soc\//  |-/' >>
devicetree.txt
 echo "" >> devicetree.txt
 cat devicetree.txt
```

```
[devicetree]
|-[dcmi]
| |-dcmi@4c006000
| |-dcmi@4c006000/compatible
| |-dcmi@4c006000/clocks
| |-dcmi@4c006000/resets
| |-dcmi@4c006000/pinctrl-1
| |-dcmi@4c006000/port
| |-dcmi@4c006000/port/endpoint
| |-dcmi@4c006000/port/endpoint/hsync-active
| |-dcmi@4c006000/port/endpoint/vsync-active
| |-dcmi@4c006000/port/endpoint/remote-endpoint
| |-dcmi@4c006000/port/endpoint/bus-width
| |-dcmi@4c006000/port/endpoint/pclk-sample
| |-dcmi@4c006000/port/endpoint/phandle
| |-dcmi@4c006000/port/endpoint/linux,phandle
| |-dcmi@4c006000/port/endpoint/name
| |-dcmi@4c006000/port/name
| |-dcmi@4c006000/clock-names
| |-dcmi@4c006000/status
| |-dcmi@4c006000/interrupts
| |-dcmi@4c006000/dma-names
| |-dcmi@4c006000/phandle
| |-dcmi@4c006000/reg
| |-dcmi@4c006000/pinctrl-0
| |-dcmi@4c006000/dmas
| |-dcmi@4c006000/linux,phandle
| |-dcmi@4c006000/name
| |-dcmi@4c006000/pinctrl-names
| |-pin-controller/dcmi-sleep@0
| |-pin-controller/dcmi-sleep@0/pins
| |-pin-controller/dcmi-sleep@0/pins/pinmux
| |-pin-controller/dcmi-sleep@0/pins/name
| |-pin-controller/dcmi-sleep@0/phandle
| |-pin-controller/dcmi-sleep@0/linux,phandle
```

```
| |-pin-controller/dcmi-sleep@0/name
| |-pin-controller/dcmi@0
| |-pin-controller/dcmi@0/pins
| |-pin-controller/dcmi@0/pins/pinmux
| |-pin-controller/dcmi@0/pins/bias-disable
| |-pin-controller/dcmi@0/pins/name
| |-pin-controller/dcmi@0/phandle
| |-pin-controller/dcmi@0/linux,phandle
| |-pin-controller/dcmi@0/name
|
|-[camera:ovti,ov5640]
  |-i2c@40013000/camera@3c
  |-i2c@40013000/camera@3c/compatible
  |-i2c@40013000/camera@3c/powerdown-gpios
  |-i2c@40013000/camera@3c/DOVDD-supply
  |-i2c@40013000/camera@3c/clocks
  |-i2c@40013000/camera@3c/rotation
  |-i2c@40013000/camera@3c/port
  |-i2c@40013000/camera@3c/port/endpoint
  |-i2c@40013000/camera@3c/port/endpoint/hsync-active
  |-i2c@40013000/camera@3c/port/endpoint/vsync-active
  |-i2c@40013000/camera@3c/port/endpoint/remote-endpoint
  |-i2c@40013000/camera@3c/port/endpoint/bus-width
  |-i2c@40013000/camera@3c/port/endpoint/pclk-sample
  |-i2c@40013000/camera@3c/port/endpoint/phandle
  |-i2c@40013000/camera@3c/port/endpoint/data-shift
  |-i2c@40013000/camera@3c/port/endpoint/linux,phandle
  |-i2c@40013000/camera@3c/port/endpoint/name
  |-i2c@40013000/camera@3c/port/name
  |-i2c@40013000/camera@3c/clock-names
  |-i2c@40013000/camera@3c/status
  |-i2c@40013000/camera@3c/reset-gpios
  |-i2c@40013000/camera@3c/phandle
  |-i2c@40013000/camera@3c/reg
  |-i2c@40013000/camera@3c/linux,phandle
  |-i2c@40013000/camera@3c/name
```

5.2 How to trace

5.2.1 V4L2 userland API tracing

Tracing of V4L2 userland API[2] can be enabled using command:

```
Board $> echo 0x3 >  /sys/devices/platform/soc/*.dcmi/video4linux/video*/dev_debug
```

Traces are output in kernel log buffer:

```
Board $> dmesg
```

```
[10130.641469] video0: VIDIOC_TRY_FMT: type=vid-cap, width=640, height=480,
pixelformat=YUYV, field=none, bytesperline=
1280, sizeimage=614400, colorspace=8, flags=0x0, ycbcr_enc=1, quantization=1, xfer_func=2
[10130.641550] video0: VIDIOC_S_FMT: type=vid-cap, width=640, height=480,
pixelformat=YUYV, field=none, bytesperline=12
80, sizeimage=614400, colorspace=8, flags=0x0, ycbcr_enc=1, quantization=1, xfer_func=2
[10130.641597] video0: VIDIOC_G_PARM: type=vid-cap, capability=0x1000, capturemode=0x0,
timeperframe=1/30, extendedmode
=0, readbuffers=2
```

```
[10130.641638] video0: VIDIOC_G_PARM: type=vid-cap, capability=0x1000, capturemode=0x0,
timeperframe=1/30, extendedmode
=0, readbuffers=2
[10130.641681] video0: VIDIOC_S_PARM: type=vid-cap, capability=0x1000, capturemode=0x0,
timeperframe=1/30, extendedmode
=0, readbuffers=2
[10130.641740] video0: VIDIOC_G_CTRL: error -22: id=0x980927, value=0
[10130.642770] video0: VIDIOC_REQBUFS: count=0, type=vid-cap, memory=mmap
[10130.642819] video0: VIDIOC_CREATE_BUFS: index=0, count=0, memory=mmap, type=vid-cap,
width=640, height=480, pixelfor
mat=YUYV, field=none, bytesperline=1280, sizeimage=614400, colorspace=8, flags=0x0,
ycbcr_enc=1, quantization=1, xfer_f
unc=2
[10130.658541] video0: VIDIOC_G_CTRL: error -22: id=0x980927, value=0
[10130.662770] video0: VIDIOC_REQBUFS: count=3, type=vid-cap, memory=mmap
[10130.662852] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=0, type=vid-cap,
flags=0x00002000, field=any, sequence=
0, memory=mmap, bytesused=0, offset/userptr=0x0, length=614400
[10130.662892] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10130.662917] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=1, type=vid-cap,
flags=0x00002000, field=any, sequence=
0, memory=mmap, bytesused=0, offset/userptr=0x96000, length=614400
[10130.662952] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10130.662967] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=2, type=vid-cap,
flags=0x00002000, field=any, sequence=
0, memory=mmap, bytesused=0, offset/userptr=0x12c000, length=614400
[10130.663002] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10130.666880] video0: VIDIOC_STREAMON: type=vid-cap
[10130.857484] video0: VIDIOC_CREATE_BUFS: index=3, count=1, memory=mmap, type=vid-cap,
width=640, height=480, pixelfor
mat=YUYV, field=none, bytesperline=1280, sizeimage=614400, colorspace=8, flags=0x0,
ycbcr_enc=1, quantization=1, xfer_f
unc=2
[10130.857585] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=3, type=vid-cap,
flags=0x00002000, field=any, sequence=
0, memory=mmap, bytesused=0, offset/userptr=0x1c2000, length=614400
[10130.857627] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10131.022069] video0: VIDIOC_STREAMOFF: type=vid-cap
```

5.2.2 V4L2 core framework tracing

Tracing of the V4L2 core framework[3] can be enabled using commands:

```
Board $> echo 0x3 > /sys/module/videobuf2_core/parameters/debug
Board $> echo 0x3 > /sys/module/videobuf2_v4l2/parameters/debug
```

Traces are output in kernel log buffer:

```
Board $> dmesg
```

```
[11875.487933] vb2-core: __setup_offsets: buffer 0, plane 0 offset 0x00000000
[11875.501731] vb2-core: __setup_offsets: buffer 1, plane 0 offset 0x001fb000
[11875.514901] vb2-core: __setup_offsets: buffer 2, plane 0 offset 0x003f6000
[11875.532298] vb2-core: __setup_offsets: buffer 3, plane 0 offset 0x005f1000
[11875.540019] vb2-core: __vb2_queue_alloc: allocated 4 buffers, 1 plane(s) each
[11875.563689] vb2_dc_mmap: mapped dma addr 0xf1900000 at 0xb4f05000, size 2076672
[11875.571174] vb2-core: vb2_mmap: buffer 0, plane 0 successfully mapped
[11875.589656] vb2-core: vb2_core_qbuf: qbuf of buffer 0 succeeded
[11875.595684] vb2_dc_mmap: mapped dma addr 0xf1b00000 at 0xb4d0a000, size 2076672
[11875.603062] vb2-core: vb2_mmap: buffer 1, plane 0 successfully mapped
```

```
[11875.609668] vb2-core: vb2_core_qbuf: qbuf of buffer 1 succeeded
[11875.615642] vb2_dc_mmap: mapped dma addr 0xf1d00000 at 0xb4b0f000, size 2076672
[11875.623016] vb2-core: vb2_mmap: buffer 2, plane 0 successfully mapped
[11875.629628] vb2-core: vb2_core_qbuf: qbuf of buffer 2 succeeded
[11875.635617] vb2_dc_mmap: mapped dma addr 0xf1f00000 at 0xb4914000, size 2076672
[11875.642952] vb2-core: vb2_mmap: buffer 3, plane 0 successfully mapped
[11875.649715] vb2-core: vb2_core_qbuf: qbuf of buffer 3 succeeded
[11875.734058] vb2-core: vb2_core_streamon: successful
[11875.961291] vb2-core: vb2_buffer_done: done processing on buffer 0, state: 6
[11875.967036] vb2-core: vb2_core_dqbuf: returning done buffer
[11875.972437] vb2-core: vb2_core_dqbuf: dqbuf of buffer 0, with state 0
[11876.094639] vb2-core: vb2_buffer_done: done processing on buffer 1, state: 6
[11876.100367] vb2-core: vb2_core_dqbuf: returning done buffer
[11876.105788] vb2-core: vb2_core_dqbuf: dqbuf of buffer 1, with state 0
```

5.2.3 DCMI V4L2 kernel driver tracing

DCMI dynamic debug traces[7] can be enabled using command:

```
Board $> echo "module stm32_dcmi +p" > /sys/kernel/debug/dynamic_debug/control
```

Here is an example with a 5Mp jpeg capture:

```
Board $> gst-launch-1.0 v4l2src ! image/jpeg, width=2592, height=1944 ! fakesink
Board $> dmesg
```

```
[12706.715949] stm32-dcmi 4c006000.dcmi: Sensor format set to 0x4001 2592x1944
[12706.721548] stm32-dcmi 4c006000.dcmi: Buffer format set to JPEG 2592x1944
[12707.365947] stm32-dcmi 4c006000.dcmi: Sensor format set to 0x4001 2592x1944
[12707.371551] stm32-dcmi 4c006000.dcmi: Buffer format set to JPEG 2592x1944
[12707.437537] stm32-dcmi 4c006000.dcmi: Sensor format set to 0x4001 2592x1944
[12707.443042] stm32-dcmi 4c006000.dcmi: Buffer format set to JPEG 2592x1944
[12707.459767] stm32-dcmi 4c006000.dcmi: Setup queue, count=4, size=5038848
[12707.518914] stm32-dcmi 4c006000.dcmi: buffer[0] phy=0xf1900000 size=5038848
[12707.526068] stm32-dcmi 4c006000.dcmi: buffer[1] phy=0xf1e00000 size=5038848
[12707.533299] stm32-dcmi 4c006000.dcmi: buffer[2] phy=0xf2300000 size=5038848
[12707.541456] stm32-dcmi 4c006000.dcmi: buffer[3] phy=0xf2800000 size=5038848
[12707.551443] stm32-dcmi 4c006000.dcmi: Start streaming, starting capture
[12707.820885] stm32-dcmi 4c006000.dcmi: buffer[0] done seq=0, bytesused=499936
[12708.087436] stm32-dcmi 4c006000.dcmi: buffer[1] done seq=1, bytesused=447472
[12708.306415] stm32-dcmi 4c006000.dcmi: Stop streaming, errors=0 (overrun=0), buffers=2
[12708.319095] stm32-dcmi 4c006000.dcmi: Start streaming, starting capture
[12708.333571] stm32-dcmi 4c006000.dcmi: Stop streaming, errors=0 (overrun=0), buffers=0
```

5.3 How to debug

5.3.1 Errors

Errors are unconditionnaly traced in kernel log:

```
Board $> dmesg
[   87.233672] stm32-dcmi 4c006000.dcmi: Some errors found while streaming: errors=1
(overrun=1), buffers=24
```

5.3.2 Memory tracking

Frames require large chunks of contiguous memory, they are allocated by V4L2 framework through DMA backend. Those allocations can be traced using:

```
Board $> echo "module dma_contiguous +p" > /sys/kernel/debug/dynamic_debug/control
Board $> echo "module videobuf2_dma_contig +p" > /sys/kernel/debug/dynamic_debug/control
```

Here is the trace after a VGA preview

```
[11311.617688] vb2_dc_mmap: mapped dma addr 0xf1900000 at 0xb3b6a000, size 614400
[11311.617986] vb2_dc_mmap: mapped dma addr 0xf1a00000 at 0xb3ad4000, size 614400
[11311.618071] vb2_dc_mmap: mapped dma addr 0xf1b00000 at 0xb3a3e000, size 614400
[11311.764146] vb2_dc_mmap: mapped dma addr 0xf1c00000 at 0xb307c000, size 614400
```

4 frames of VGA YUV422 frames: 640x480x2=614400 bytes

6 Source code location

6.1 User space

- V4L2 utilities source code
- V4L2 libraries source code
- Yavta source code

6.2 Kernel space

- V4L2 core source code
- stm32-dcmi V4L2 driver source code
- i2c camera sensor V4L2 drivers source code

7 References

- Information about V4L2 Linux kernel framework on wikipedia.
- 2.02.1 Linux Media Infrastructure userspace API » Part I - Video for Linux API
- 3.03.1 Media subsystem kernel internal API » 1. Video4Linux devices
- MB1379 camera daughter board
- STM32MP157x-EV1 Evaluation board CN7 Camera sensor connector
- STM32MP15 evaluation board
- How to use the kernel dynamic debug

Linux® is a registered trademark of Linus Torvalds.

Video 4 Linux version 2

Digital Camera Memory Interface

Application programming interface

Inter-Integrated Circuit (Bi-directional 2-wire bus standard for efficient inter-IC control.)

Direct Memory Access