

# PXA-Camera Host Driver

Author: Robert Jarzmik <[robert.jarzmik@free.fr](mailto:robert.jarzmik@free.fr)>

## Constraints

- Image size for YUV422P format All YUV422P images are enforced to have width x height % 16 = 0. This is due to DMA constraints, which transfers only planes of 8 byte multiples.

## Global video workflow

- QCI stopped Initially, the QCI interface is stopped. When a buffer is queued (pxa\_videobuf\_ops->buf\_queue), the QCI starts.
- QCI started More buffers can be queued while the QCI is started without halting the capture. The new buffers are "appended" at the tail of the DMA chain, and smoothly captured one frame after the other.

Once a buffer is filled in the QCI interface, it is marked as "DONE" and removed from the active buffers list. It can be then requested or dequeued by userland application.

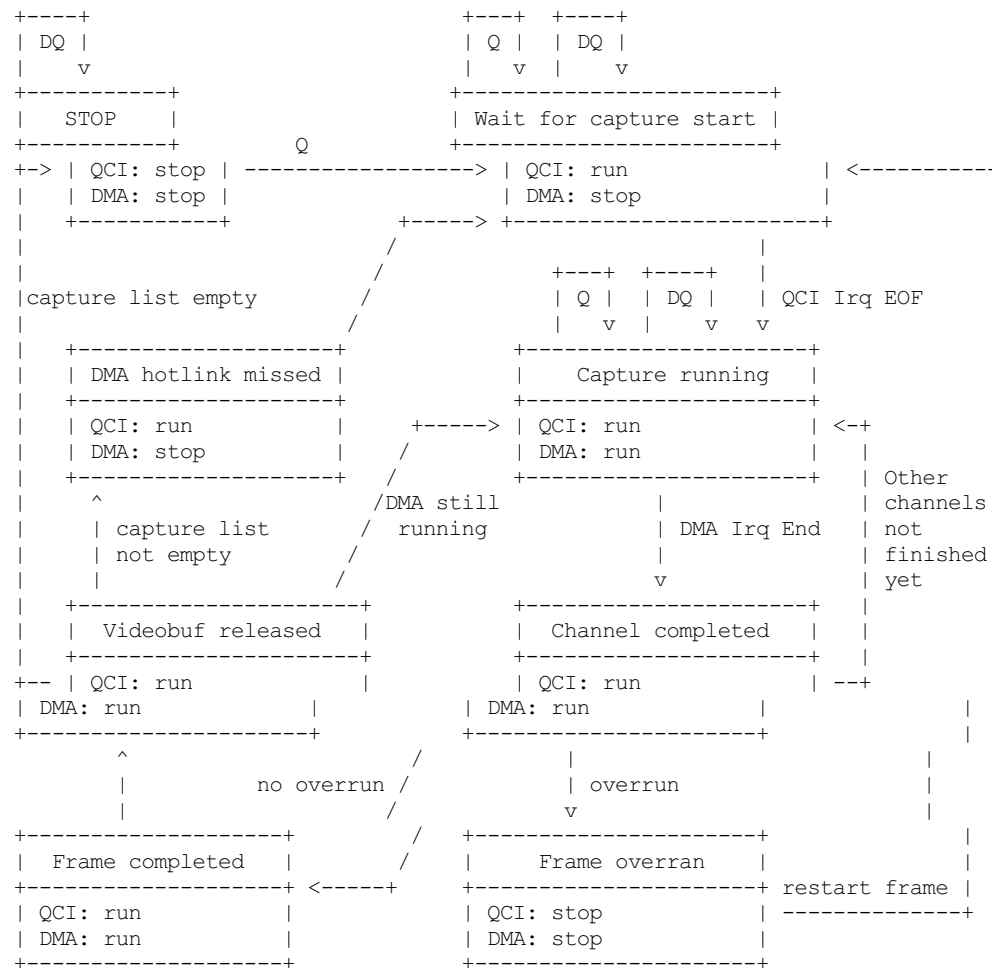
Once the last buffer is filled in, the QCI interface stops.

- Capture global finite state machine schema

**System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\linux-master [Documentation] [driver-api] [media] [drivers]pxa\_camera.rst, line 37)**

Cannot analyze code. No Pygments lexer found for "none".

.. code-block:: none



- arrow "DQ" means : a buffer was dequeued
- "QCI: stop" means the QCI interface is not enabled
- "DMA: stop" means all 3 DMA channels are stopped
- "DMA: run" means at least 1 DMA channel is still running

## DMA usage

### a. DMA flow

- first buffer queued for capture Once a first buffer is queued for capture, the QCI is started, but data transfer is not started. On "End Of Frame" interrupt, the irq handler starts the DMA chain.
- capture of one videobuffer The DMA chain starts transferring data into videobuffer RAM pages. When all pages are transferred, the DMA irq is raised on "ENDINTR" status
- finishing one videobuffer The DMA irq handler marks the videobuffer as "done", and removes it from the active running queue Meanwhile, the next videobuffer (if there is one), is transferred by DMA
- finishing the last videobuffer On the DMA irq of the last videobuffer, the QCI is stopped.

### b. DMA prepared buffer will have this structure

**System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\linux-master [Documentation] [driver-api] [media] [drivers]pxa\_camera.rst, line 107)**

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

+-----+-----+-----+-----+
| desc-sg[0] | ... | desc-sg[last] | finisher/linker |
+-----+-----+-----+-----+
```

This structure is pointed by dma->sg\_cpu. The descriptors are used as follows:

- desc-sg[i]: i-th descriptor, transferring the i-th sg element to the video buffer scatter gather
- finisher: has ddadr=DADDR\_STOP, dcmd=ENDIRQEN
- linker: has ddadr= desc-sg[0] of next video buffer, dcmd=0

For the next schema, let's assume d0=desc-sg[0] .. dN=desc-sg[N], "f" stands for finisher and "l" for linker. A typical running chain is :

**System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\linux-master [Documentation] [driver-api] [media] [drivers]pxa\_camera.rst, line 125)**

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

      Videobuffer 1      Videobuffer 2
+-----+-----+-----+ +-----+-----+-----+
| d0 | .. | dN | l | | d0 | .. | dN | f |
+-----+-----+-----+ ^-----+-----+-----+
                        |
                        +-----+
```

After the chaining is finished, the chain looks like :

**System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\linux-master [Documentation] [driver-api] [media] [drivers]pxa\_camera.rst, line 136)**

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

      Videobuffer 1      Videobuffer 2      Videobuffer 3
+-----+-----+-----+ +-----+-----+-----+ +-----+-----+-----+
| d0 | .. | dN | l | | d0 | .. | dN | l | | d0 | .. | dN | f |
+-----+-----+-----+ ^-----+-----+-----+ ^-----+-----+-----+
                        |               |               |
                        +-----+       +-----+
                                   new_link
```

c. DMA hot chaining timeslice issue

As DMA chaining is done while DMA `_is_` running, the linking may be done while the DMA jumps from one Videobuffer to another. On the schema, that would be a problem if the following sequence is encountered :

- DMA chain is Videobuffer1 + Videobuffer2
- `pxa_videobuf_queue()` is called to queue Videobuffer3
- DMA controller finishes Videobuffer2, and DMA stops

**System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master] [Documentation] [driver-api] [media] [drivers]pxa\_camera.rst, line 156)**

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

=>
    Videobuffer 1      Videobuffer 2
+-----+-----+
| d0 | .. | dN | l | | d0 | .. | dN | f |
+-----+-----+
|           |           |
+-----+           +--- DMA DDADR loads DDADR_STOP
```

- `pxa_dma_add_tail_buf()` is called, the Videobuffer2 "finisher" is replaced by a "linker" to Videobuffer3 (creation of `new_link`)
- `pxa_videobuf_queue()` finishes
- the DMA irq handler is called, which terminates Videobuffer2
- Videobuffer3 capture is not scheduled on DMA chain (as it stopped !!!)

**System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\media\drivers\[linux-master] [Documentation] [driver-api] [media] [drivers]pxa\_camera.rst, line 172)**

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

    Videobuffer 1      Videobuffer 2      Videobuffer 3
+-----+-----+ +-----+-----+ +-----+-----+
| d0 | .. | dN | l | | d0 | .. | dN | l | | d0 | .. | dN | f |
+-----+-----+ ^-----+-----+ ^-----+-----+
|           |           |           |           |
+-----+           +-----+
                        new_link
DMA DDADR still is DDADR_STOP
```

- `pxa_camera_check_link_miss()` is called This checks if the DMA is finished and a buffer is still on the `pcdev->capture` list. If that's the case, the capture will be restarted, and Videobuffer3 is scheduled on DMA chain.
- the DMA irq handler finishes

#### Note

If DMA stops just after `pxa_camera_check_link_miss()` reads `DDADR()` value, we have the guarantee that the DMA irq handler will be called back when the DMA will finish the buffer, and `pxa_camera_check_link_miss()` will be called again, to reschedule Videobuffer3.