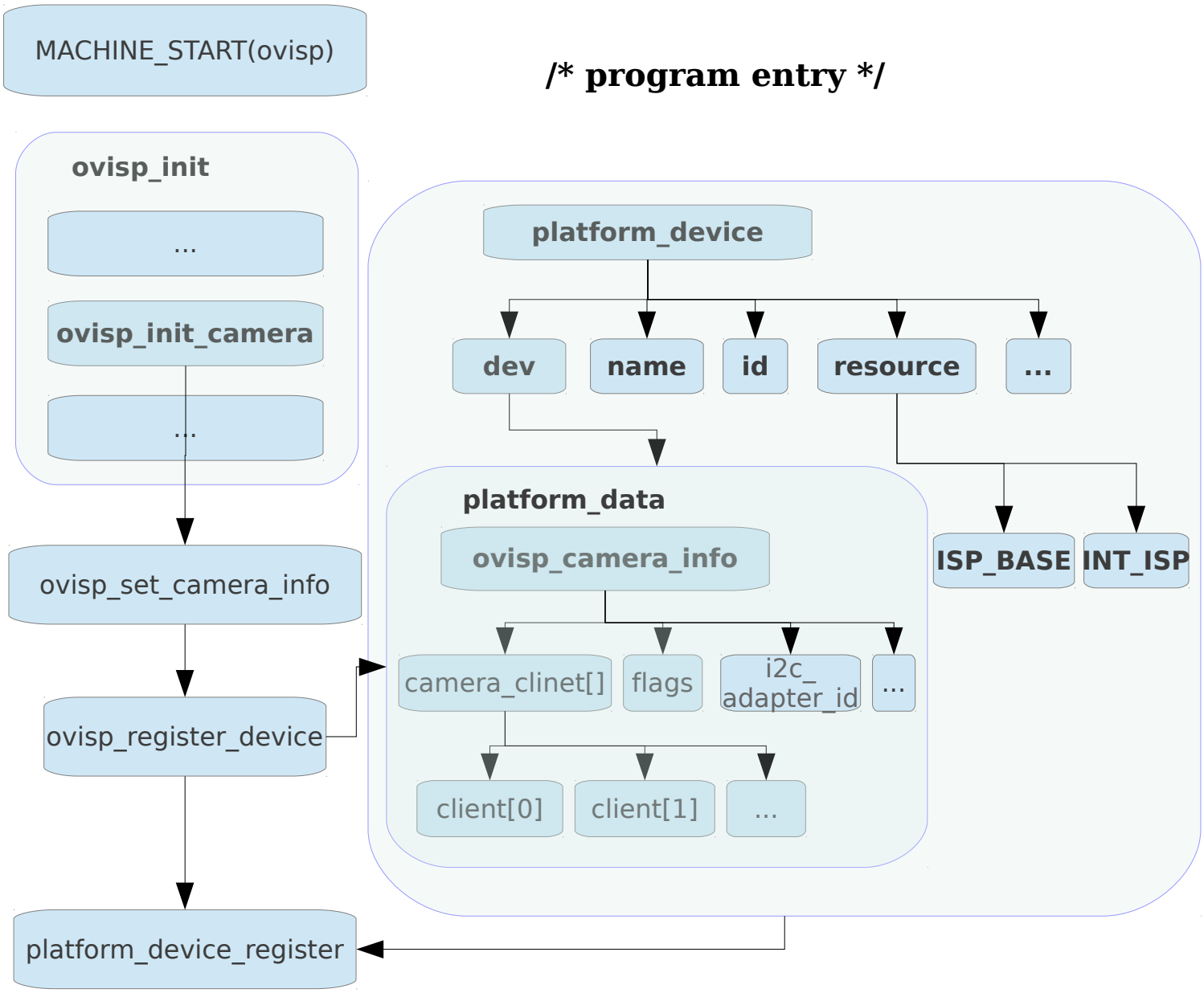
# Ovisp_Driver_Spec

## (1) register  platform_device

arch/arm/mach-xxx/board_xxx.c




add ovisp_camera_info to platform_device->dev.platform_data, add resource ISP_BASE and INT_ISP to platform_device->resource, then register platform_device

```
/*in arch/arm/mach-xxx/device.c */

/*ovisp_device_camera defined here*/

struct platform_device ovisp_device_camera = {
    .name = "ovisp-camera",
    .id = -1,
    .dev = {
       .dma_mask = &ovisp_camera_dma_mask,   /*~(u64)0*/
       .coherent_dma_mask = 0xffffffff,
    },
    .num_resources = ARRAY_SIZE(ovisp_resource_camera),
    .resource = ovisp_resource_camera,
};


/* resource_camera defined here */

static struct resource ovisp_resource_camera[] = {
    [0] = {
       .start = ISP_BASE,
       .end = ISP_BASE + 0x80000,
       .flags = IORESOURCE_MEM,
    },
    [1] = {
       .start = INT_ISP,
       .end = INT_ISP,
       .flags = IORESOURCE_IRQ,
    },
};


/*camera_clients for v4l2_subdev defined here*/

static struct ovisp_camera_client ovisp_camera_clients[] = {
#if defined(CONFIG_VIDEO_OV5647) {
       .board_info = &ov5647_board_info,
       .flags = CAMERA_CLIENT_IF_MIPI,
       .mclk_rate = 26000000, /* can be set when isp_open */
       .max_video_width = 1280,
       .max_video_height = 960,
       .power = ov5647_power,
       .reset = ov5647_reset,
   },
 #endif
 #if defined(CONFIG_VIDEO_Front_camera) {
       .board_info = ,
       .flags = CAMERA_CLIENT_IF_DVP
            | CAMERA_CLIENT_CLK_EXT
            | CAMERA_CLIENT_ISP_BYPASS,
```

```
        .mclk_parent_name =,
        .mclk_name = ,
        .mclk_rate = ,
        .power = ,
        .reset = ,
    },
#endif
};

static struct ovisp_camera_platform_data ovisp_camera_info = {
    .i2c_adapter_id = 3,
    .flags = CAMERA_USE_ISP_I2C | CAMERA_USE_HIGH_BYTE
          | CAMERA_I2C_PIO_MODE | CAMERA_I2C_STANDARD_SPEED,
    .client = ovisp_camera_clients,
    .client_num = ARRAY_SIZE(ovisp_camera_clients),
 };
```

## (2)register ovisp_camera_ driver

drivers/media/video/ovisp/ovisp_video.c

init:

**platform_driver_register(&ovisp_camera_driver);**

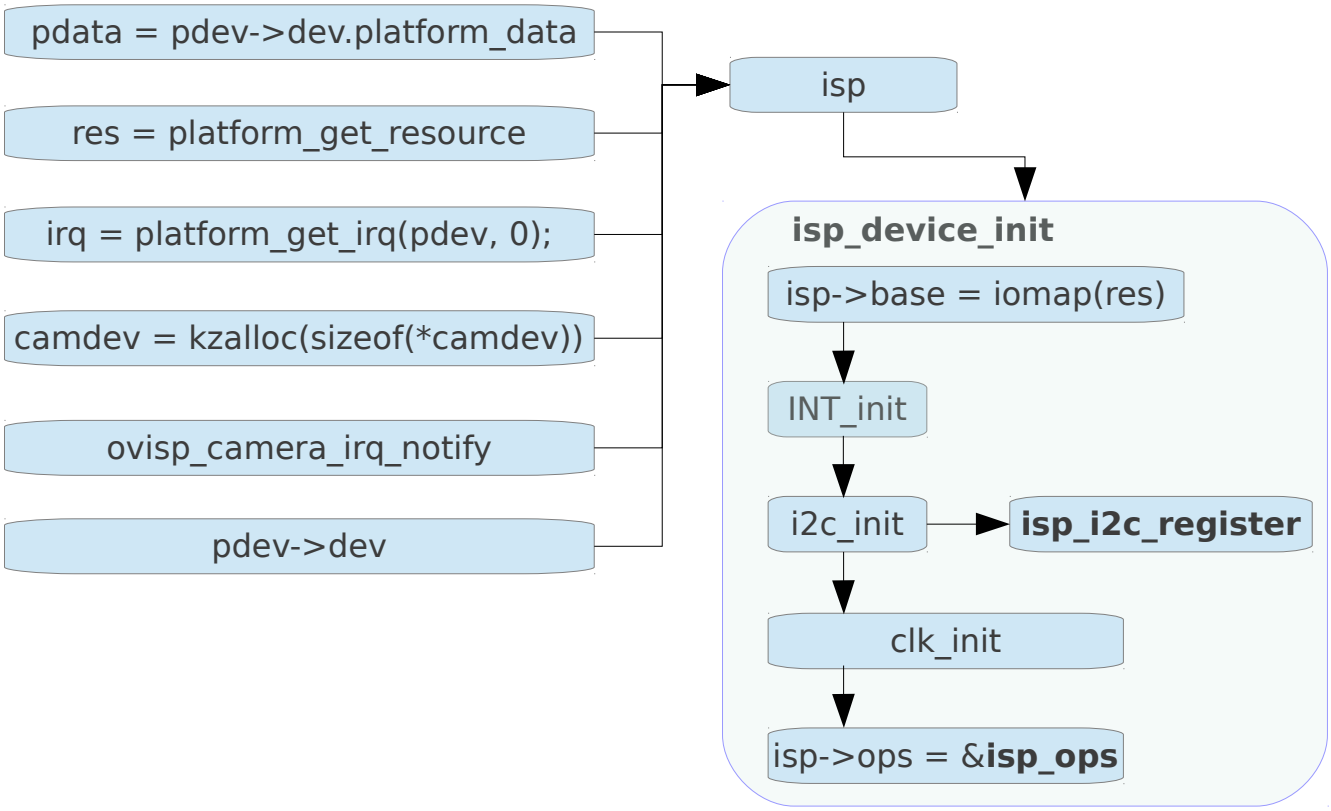
probe:

**ovisp_camera_probe(platform_device * pdev)**

/* platform_device registered in step (1), got platform_data from pdev */

|

**isp_device_init(isp);**



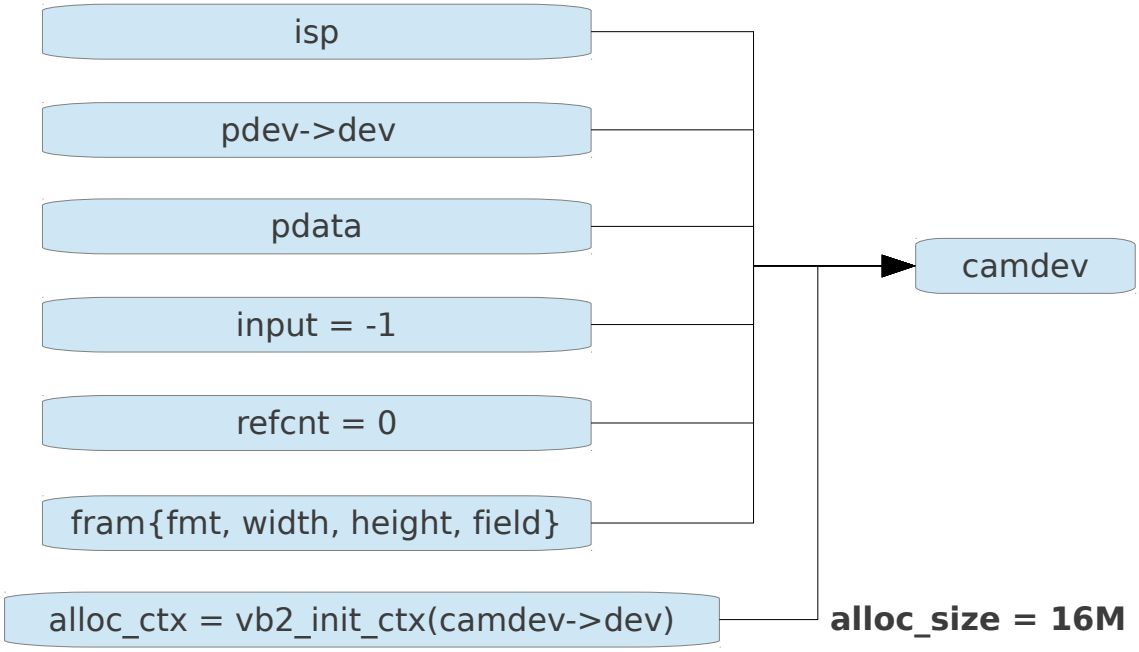

|

**v4l2_device_register(NULL, &camdev->v4l_dev);**

/* init camdev->v4l_dev , dev = NULL ,return 0, */

|

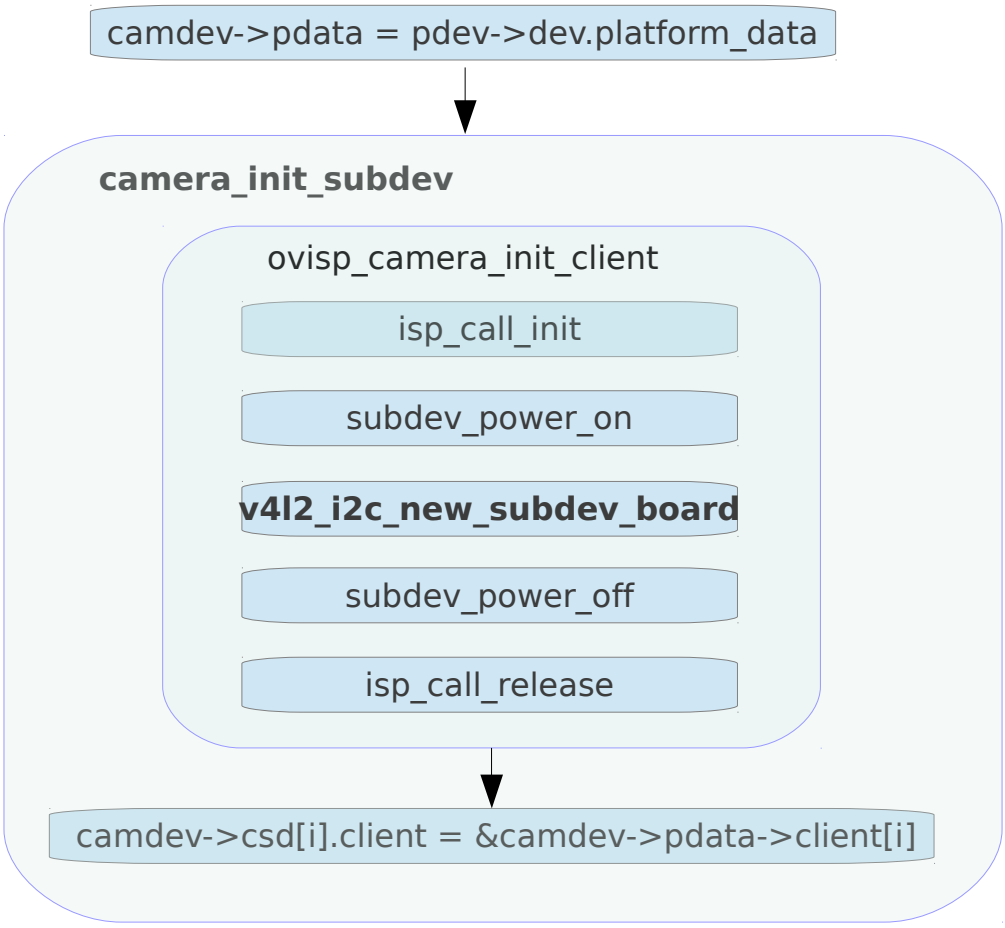
**camdev_init**




|

**camera_init_subdev(&camdev);**

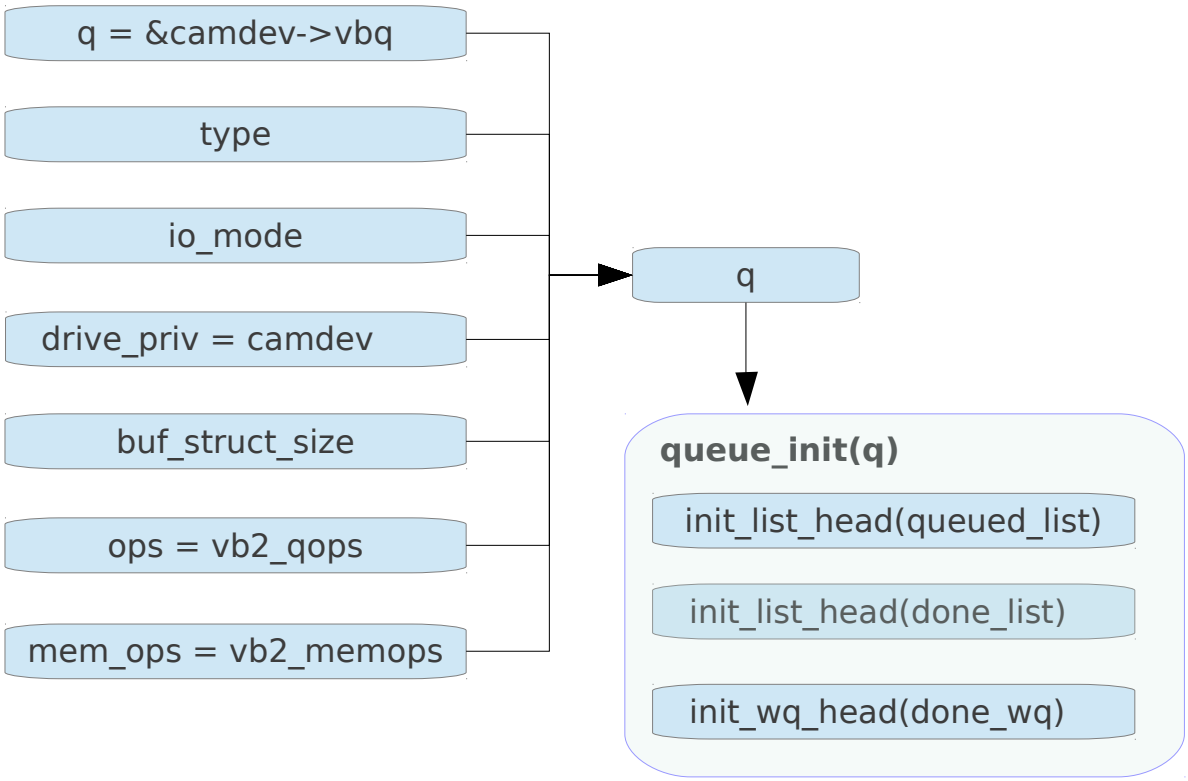
/*ovisp_camera_init_client(camdev,  &pdata->client[0/1], 0/1);*/
/*v4l2_subdev  csd->sd = v4l2_i2c_new_subdev_board() */
/* the process of v4l2_i2c_new_subdev_board described in part (5) */



|

**queue_init**

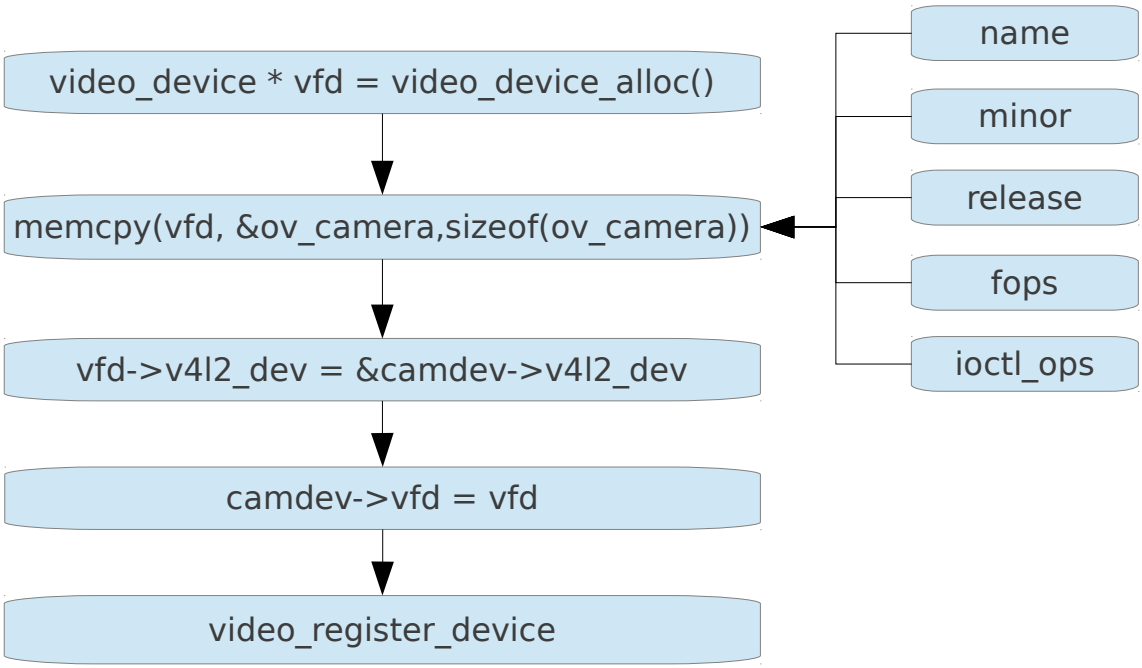
/* initialized queue info,  alloc_queue will be finished in  reqbufs */



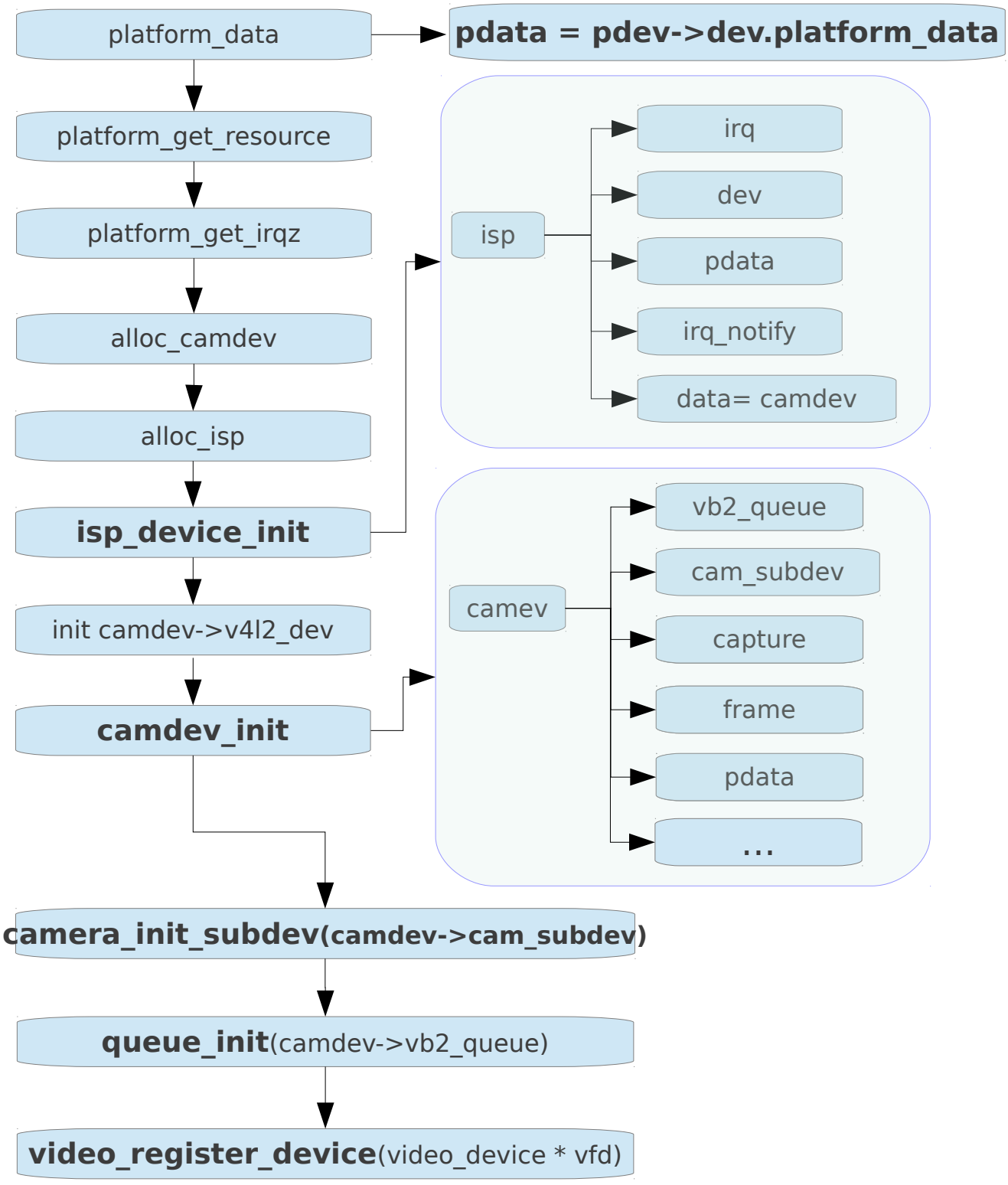

|

**video_register_device(vfd, VFL_TYPE_GRABBER, -1);**

/* register video device vfd */

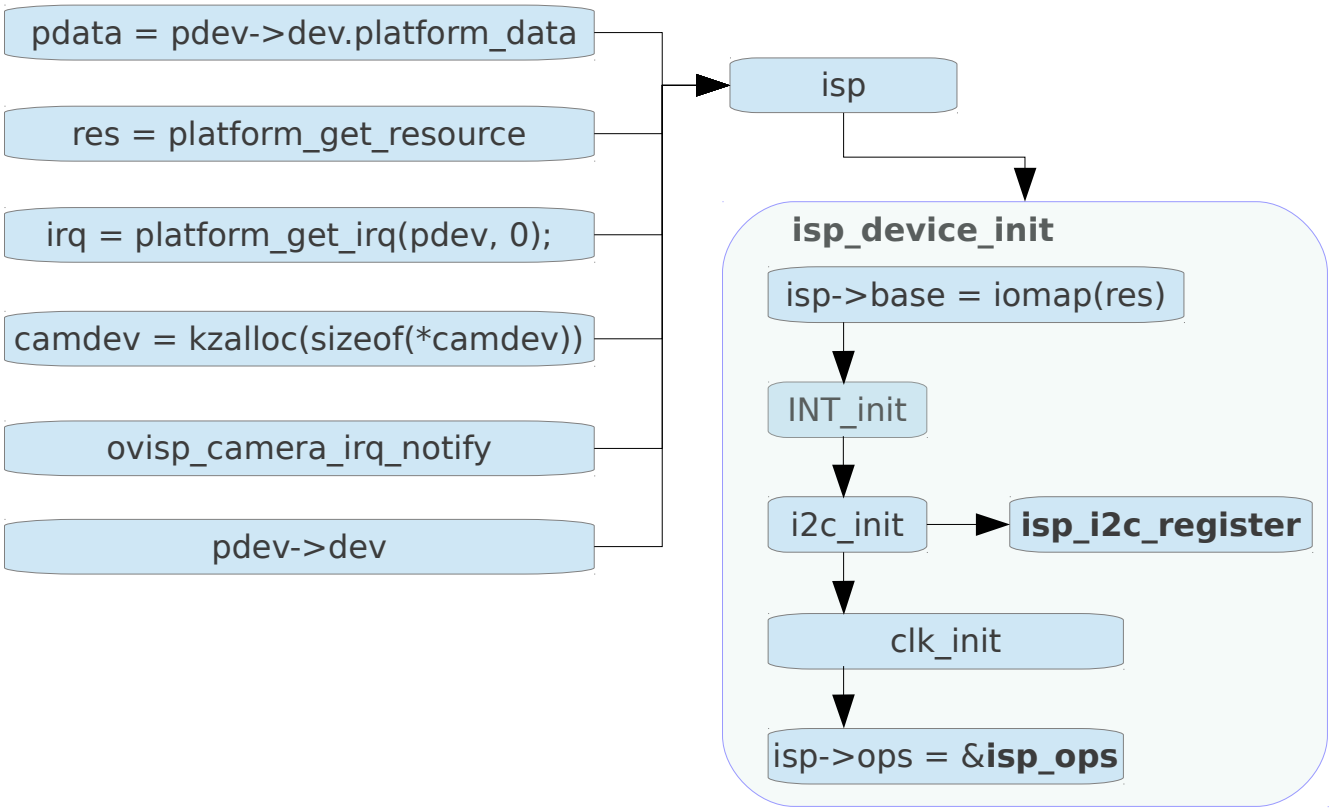


```
/* video_device */
static struct video_device ovisp_camera = {
    .name = "ovisp-camera",
    .minor = -1,
    .release = video_device_release,
    .fops = &ovisp_v4l2_fops,
    .ioctl_ops = &ovisp_v4l2_ioctl_ops,
};



/*struct platform_driver*/

static struct platform_driver ovisp_camera_driver = {
    .probe = ovisp_camera_probe,
    .remove = __exit_p(ovisp_camera_remove),
    .driver = {
       .name = "ovisp-camera",
       .owner = THIS_MODULE,
#ifdef CONFIG_PM
       .pm = &ovisp_camera_pm_ops,
#endif
    },
};
```


platform_data
pdata = pdev->dev.platform_data
platform_get_resource
platform_get_irqz
alloc_camdev
alloc_isp
isp_device_init
init camdev->v4l2_dev
camdev_init
isp
irq
dev
pdata
irq_notify
data= camdev
camev
vb2_queue
cam_subdev
capture
frame
pdata
...
camera_init_subdev(camdev->cam_subdev)
queue_init(camdev->vb2_queue)
video_register_device(video_device * vfd)

## (3)register and ioctl for ISP

/* init_isp in camera_probe*/




```
static struct resource ovisp_resource_camera[] = {
    [0] = {
        .start = ISP_BASE,
        .end = ISP_BASE + 0x80000,
        .flags = IORESOURCE_MEM,
    },
    [1] = {
        .start = INT_ISP,
        .end = INT_ISP,
        .flags = IORESOURCE_IRQ,
    },
};

static struct isp_clk_info isp_clks[ISP_CLK_NUM] = {
    {"isp_cphy_cfg_clk",    26000000,  ISP_CLK_CSI},
    {"isp_axi_clk",     156000000,  ISP_CLK_CSI},
    {"isp_p_sclk",      156000000,  ISP_CLK_MAIN | ISP_CLK_DEV},
    {"isp_sclk2",       156000000,  ISP_CLK_MAIN},
    {"isp_hclk",        200000000, ISP_CLK_MAIN},
 };
```

isp->res   get ovisp_resource_camera[0]
isp->irq   get ovisp_resource_camera[1]
isp->base get isp base addr by iomap
isp_irq registered in INT_init, irq_notify will be called in isp_irq
isp_i2c_register in i2c_init
clk_init config 5 isp_clk to choose

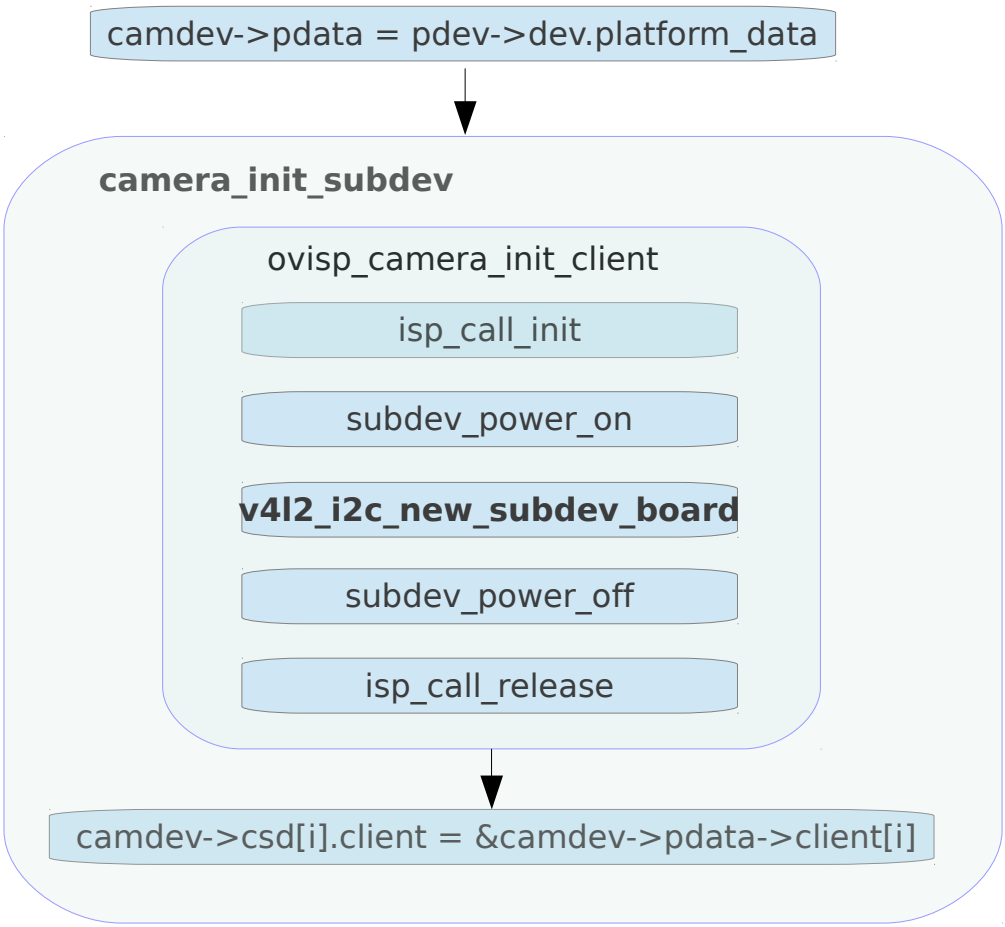
/* isp_ops used to ctrl isp by app ioctl */

```
/* struct isp_ops */
static struct isp_ops isp_ops = {
    .init = isp_init,
    .release = isp_release,
    .open = isp_open,
    .close = isp_close,
    .config = isp_config,
    .suspend = isp_suspend,
    .resume = isp_resume,
    .mclk_on = isp_mclk_on,
    .mclk_off = isp_mclk_off,
    .start_capture = isp_start_capture,
    .stop_capture = isp_stop_capture,
    .enable_capture = isp_enable_capture,
    .disable_capture = isp_disable_capture,
    .update_buffer = isp_update_buffer,
    .check_fmt = isp_check_fmt,
    .try_fmt = isp_try_fmt,
    .pre_fmt = isp_pre_fmt,
    .s_fmt = isp_s_fmt,
    .s_ctrl = isp_s_ctrl,
    .g_ctrl = isp_g_ctrl,
    .s_parm = isp_s_parm,
    .g_parm = isp_g_parm,
};
```

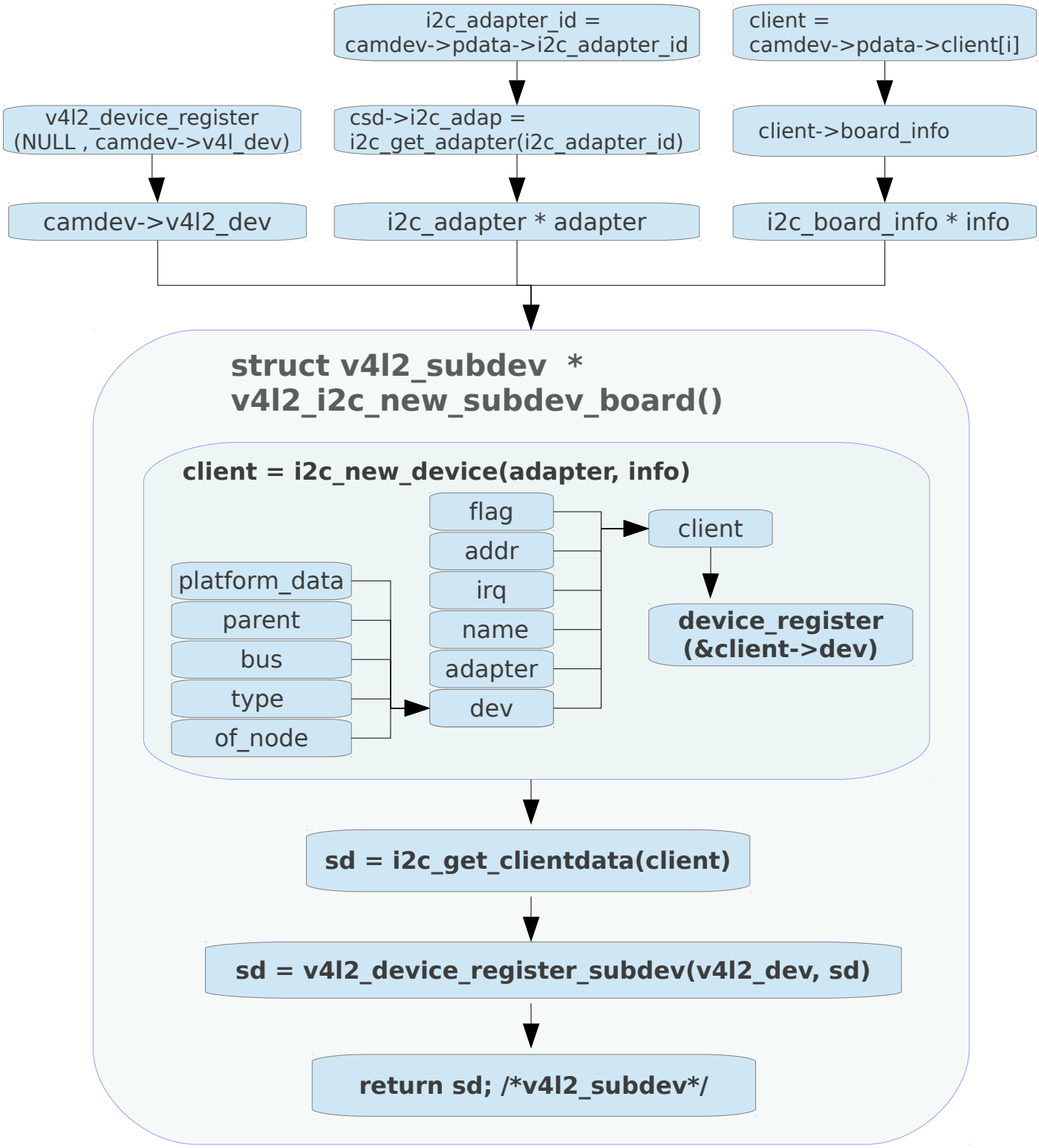
## (4)register and ioctl ov5647

drivers/media/video/ov5647.c

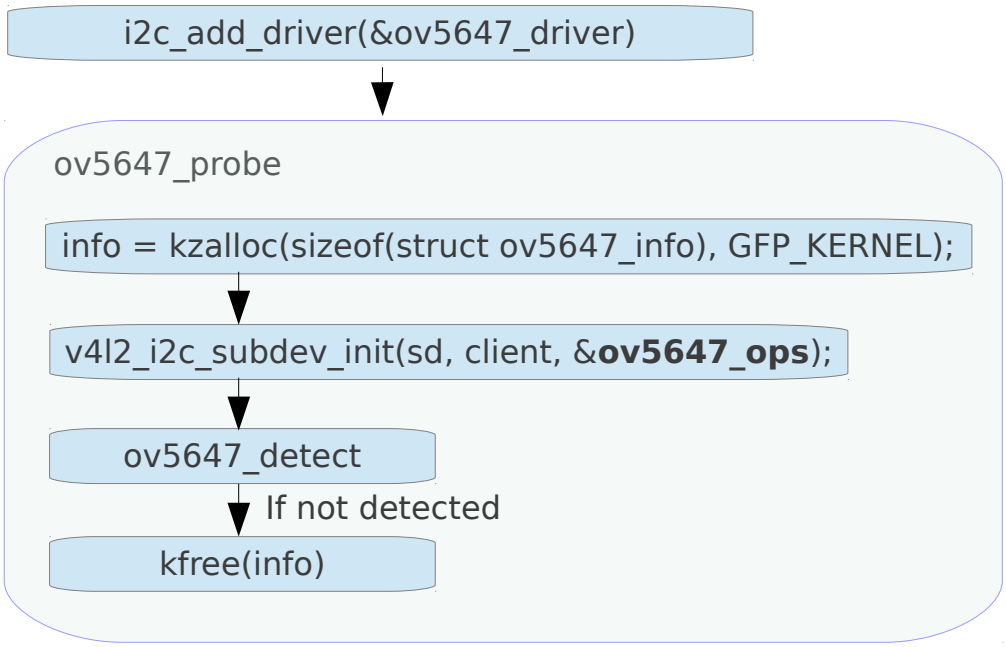
/* init_subdev process in camera_probe() */



```
csd =  &camdev->csd[index];

csd->sd = v4l2_i2c_new_subdev_board(&camdev->v4l2_dev,
        csd->i2c_adap,
        client->board_info,
        NULL);
```



/* process in ov5647 driver */




```
/* i2c driver for ov5647 */
static struct i2c_driver ov5647_driver = {
    .driver = {
        .owner  = THIS_MODULE,
        .name   = "ov5647",
    },
    .probe      = ov5647_probe,
    .remove     = ov5647_remove,
    .id_table   = ov5647_id,
};


/*ov5647_ops */
static const struct v4l2_subdev_core_ops ov5647_core_ops = {
    .g_chip_ident = ov5647_g_chip_ident,
    .g_ctrl = ov5647_g_ctrl,
    .s_ctrl = ov5647_s_ctrl,
    .queryctrl = ov5647_queryctrl,
    .reset = ov5647_reset,
    .init = ov5647_init,
 #ifdef CONFIG_VIDEO_ADV_DEBUG
    .g_register = ov5647_g_register,
    .s_register = ov5647_s_register,
 #endif
 };
```

```
static const struct v4l2_subdev_video_ops ov5647_video_ops = {
    .enum_mbus_fmt = ov5647_enum_mbus_fmt,
    .try_mbus_fmt = ov5647_try_mbus_fmt,
    .s_mbus_fmt = ov5647_s_mbus_fmt,
    .s_stream = ov5647_s_stream,
    .cropcap = ov5647_cropcap,
    .g_crop = ov5647_g_crop,
    .s_parm = ov5647_s_parm,
    .g_parm = ov5647_g_parm,
    .enum_frameintervals = ov5647_enum_frameintervals,
    .enum_framesizes = ov5647_enum_framesizes,
};

static const struct v4l2_subdev_ops ov5647_ops = {
    .core = &ov5647_core_ops,
    .video = &ov5647_video_ops,
};
```

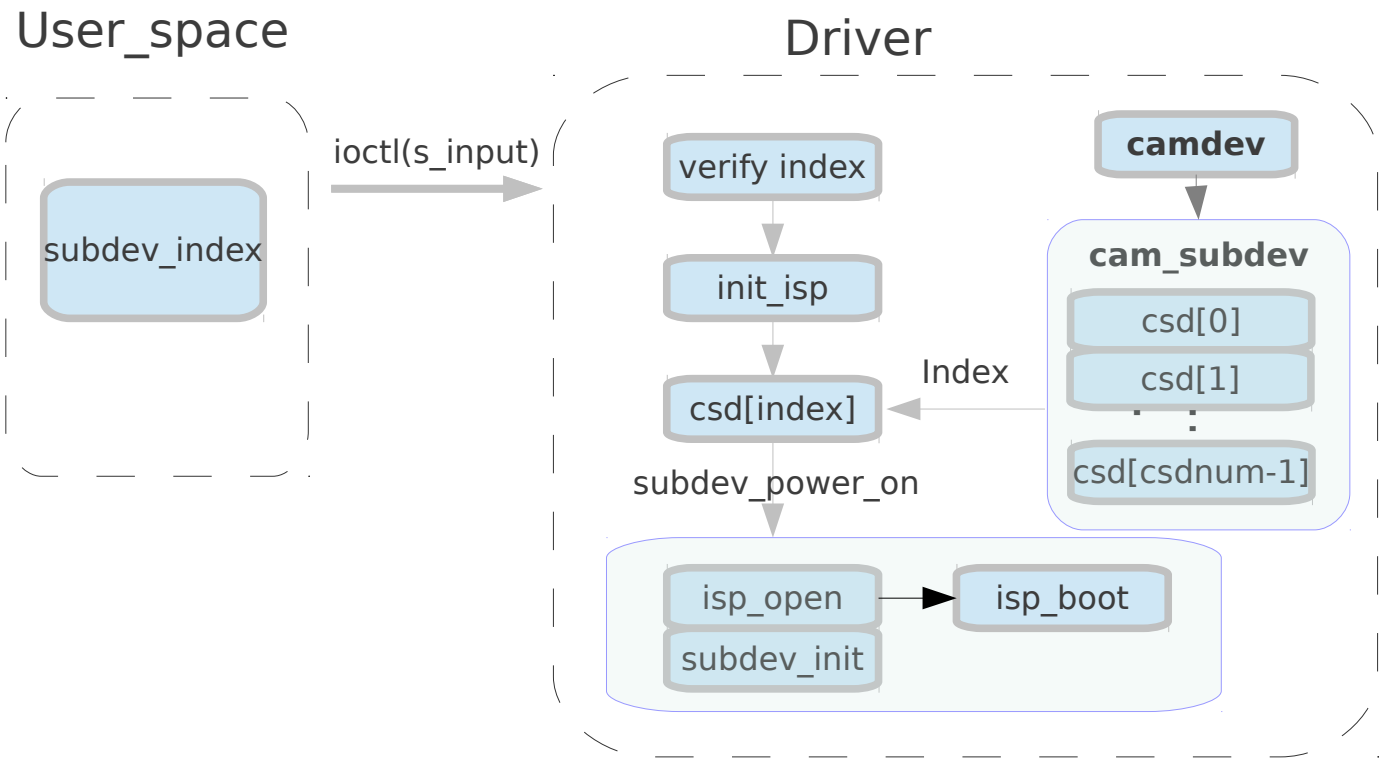
## (5)Buffer Management

**ovisp_vidioc_s_input**((struct file *file, void *priv,  unsigned int index))

para:

file:  file handle to get video_device;
priv:  have not used;
index: client index for subdev;

process:

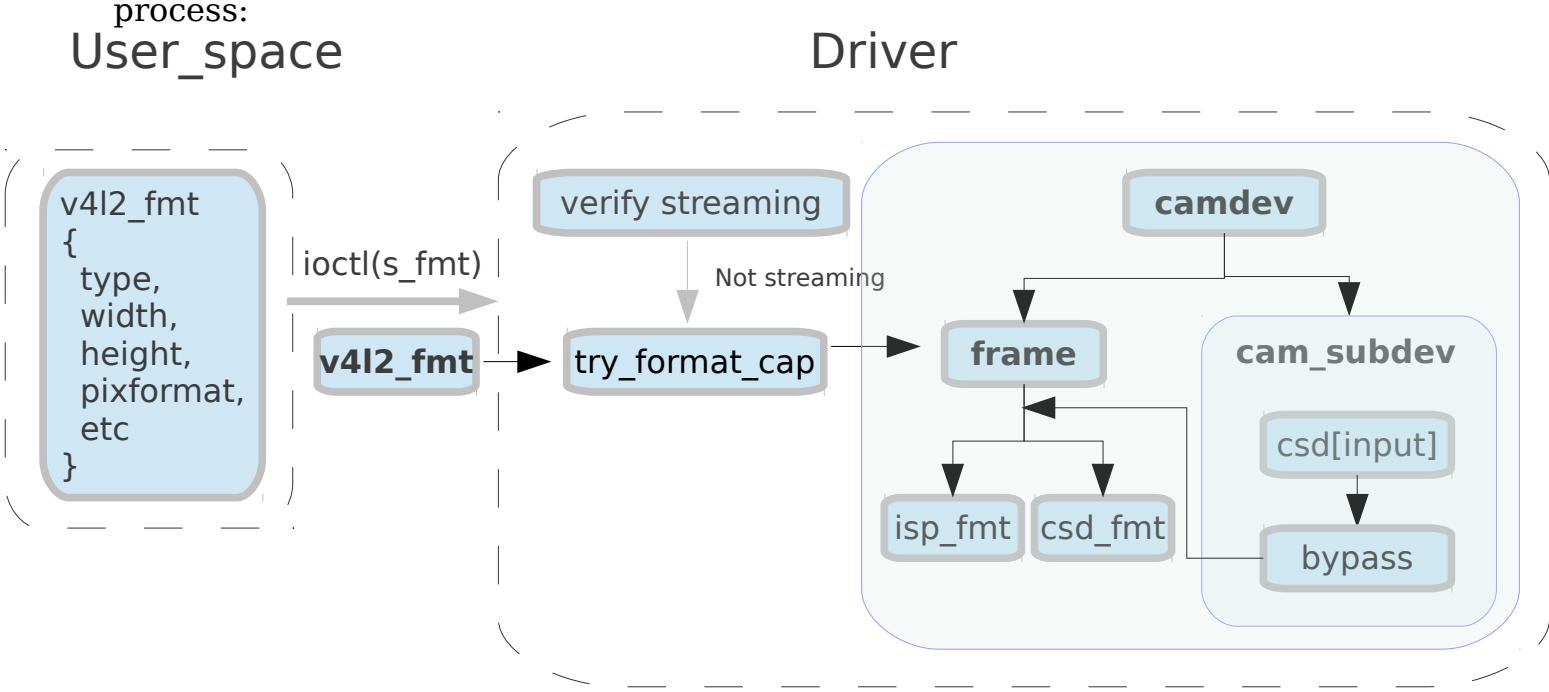


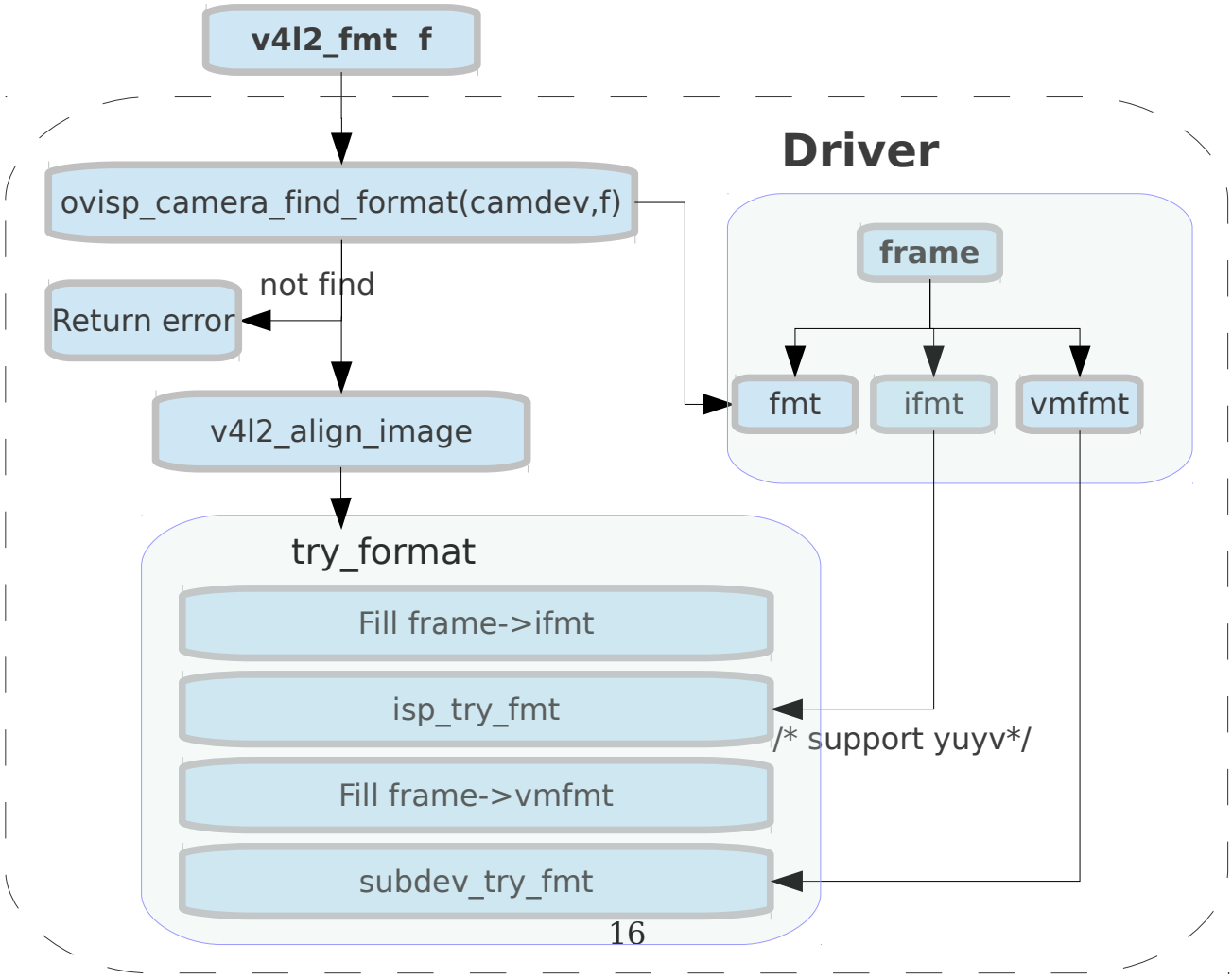
**ovisp_vidioc_s_fmt_vid_cap**(struct file *file, void *priv, struct v4l2_format *f)

para:

f: v4l2_format set by app, include v4l2_type, video image format

process:



**/* isp format */**

```
 static struct ovisp_camera_format formats[] = {
    {
       .name     = "YUV 4:2:2 packed, YCbYCr",
       .code     = V4L2_MBUS_FMT_SBGGR8_1X8,
       .fourcc   = V4L2_PIX_FMT_YUYV,
       .depth    = 16,
    },
    {
       .name     = "YUV 4:2:0 semi planar, Y/CbCr",
       .code     = V4L2_MBUS_FMT_SBGGR8_1X8,
       .fourcc   = V4L2_PIX_FMT_NV12,
       .depth    = 12,
    },
 };

 static struct ovisp_camera_format bypass_formats[] = {
    {
       .name     = "YUV 4:2:2 packed, YCbYCr",
       .code     = V4L2_MBUS_FMT_YUYV8_2X8,
       .fourcc   = V4L2_PIX_FMT_YUYV,
       .depth    = 16,
    },
 };
```

**/* ov5647 format */**

```
static struct ov5647_format_struct {
    enum v4l2_mbus_pixelcode mbus_code;
    enum v4l2_colorspace colorspace;
    struct regval_list *regs;
 } ov5647_formats[] = {
    {
       .mbus_code  = V4L2_MBUS_FMT_SBGGR8_1X8,
       .colorspace = V4L2_COLORSPACE_SRGB,
       .regs       = NULL,
    },
 };
```

**ovisp_vidioc_reqbufs**(struct file *file, void *priv, struct, v4l2_requestbuffers *p)

para:

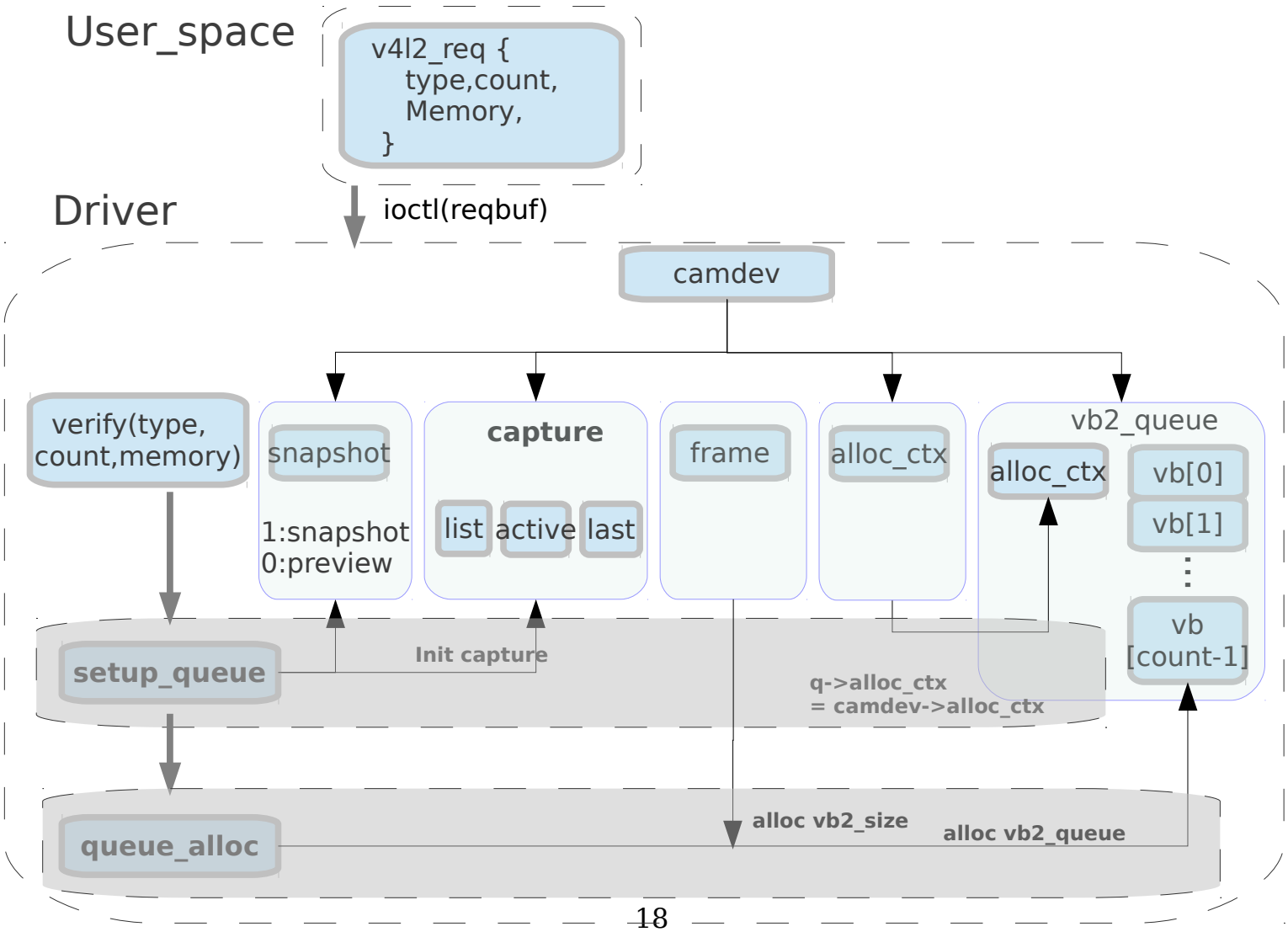
p: struct passed from userspace

functions:

1) verifies streaming parameters passed from the userspace,
2) sets up the queue,
3) negotiates number of buffers and planes per buffer with the driver to be used during streaming,
4) allocates internal buffer structures (struct vb2_buffer), according to the agreed parameters,
5) for MMAP memory type, allocates actual video memory, using the memory handling/allocation routines provispided during queue initialization

process:

vb2_reqbufs(struct vb2_queue *q, struct v4l2_requestbuffers *req)



**ovisp_vidioc_querybuf**(struct file *file, void *priv, struct v4l2_buffer *p)

para:

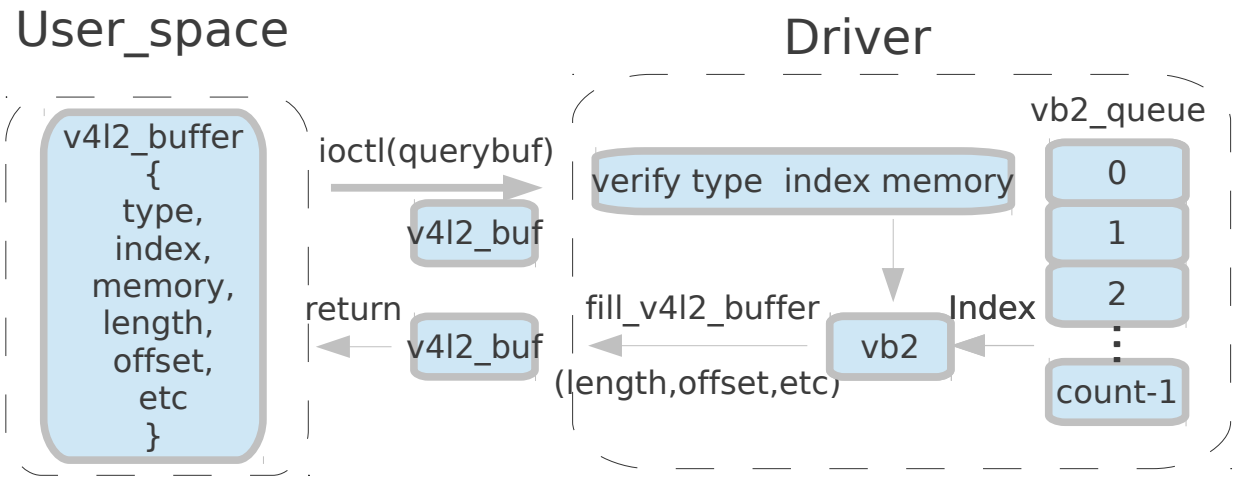
function:

verify the passed v4l2_buffer structure and fill the relevant information for the userspace.

process:

vb2_qbuf(struct vb2_queue *q, struct v4l2_buffer *b)

struct vb2_queue  q = camdev->vbq



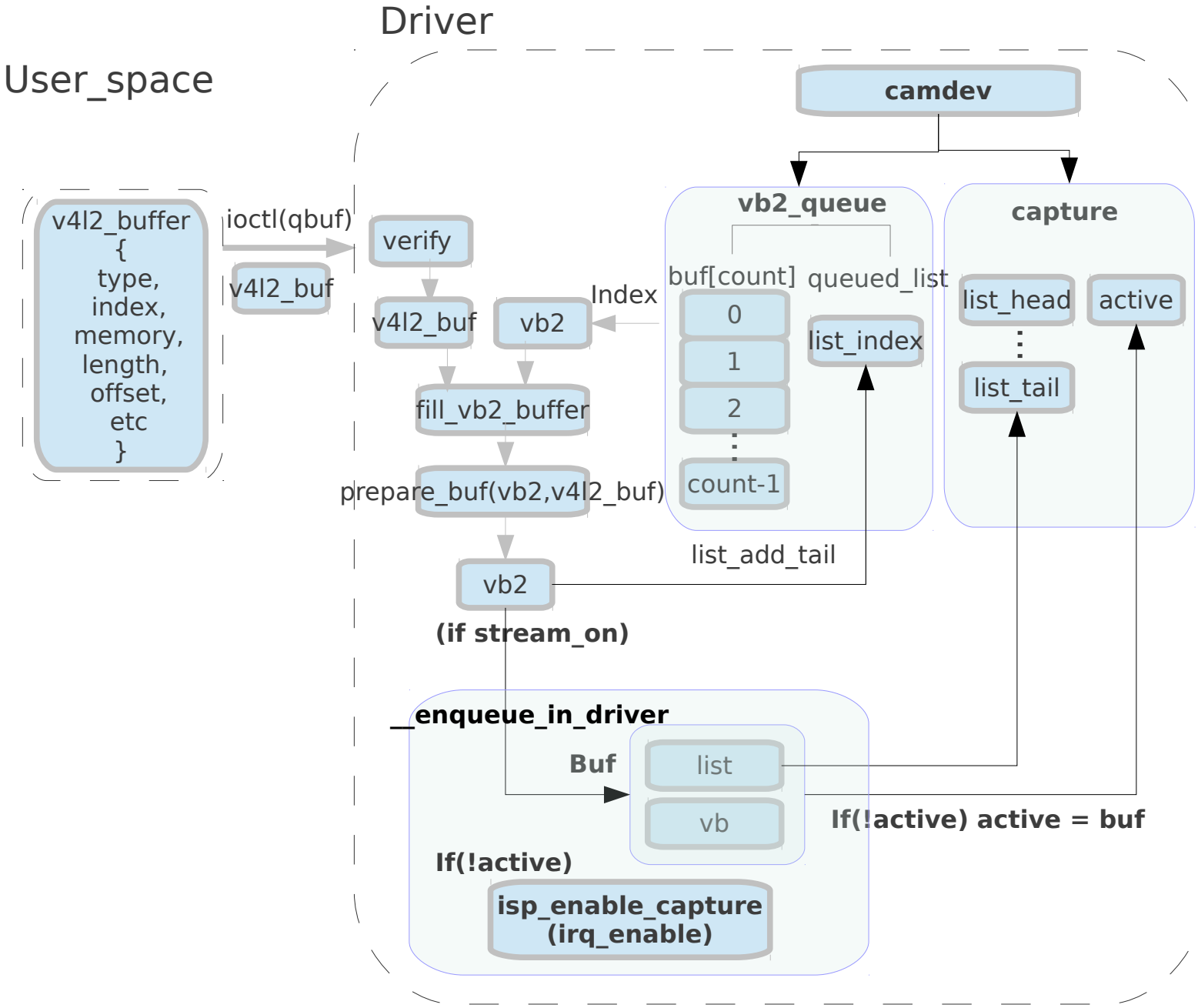
**ovisp_vidioc_qbuf**(struct file *file, void *priv, struct v4l2_buffer *p)

para:
function:

1) verifies the passed buffer,
2) calls buf_prepare callback in the driver (if provispided), in which driver-specific buffer initialization can be performed,
3) if streaming is on, queues the buffer in driver by the means of buf_queue callback for processing.

Progress:

vb2_qbuf(struct vb2_queue *q, struct v4l2_buffer *b)



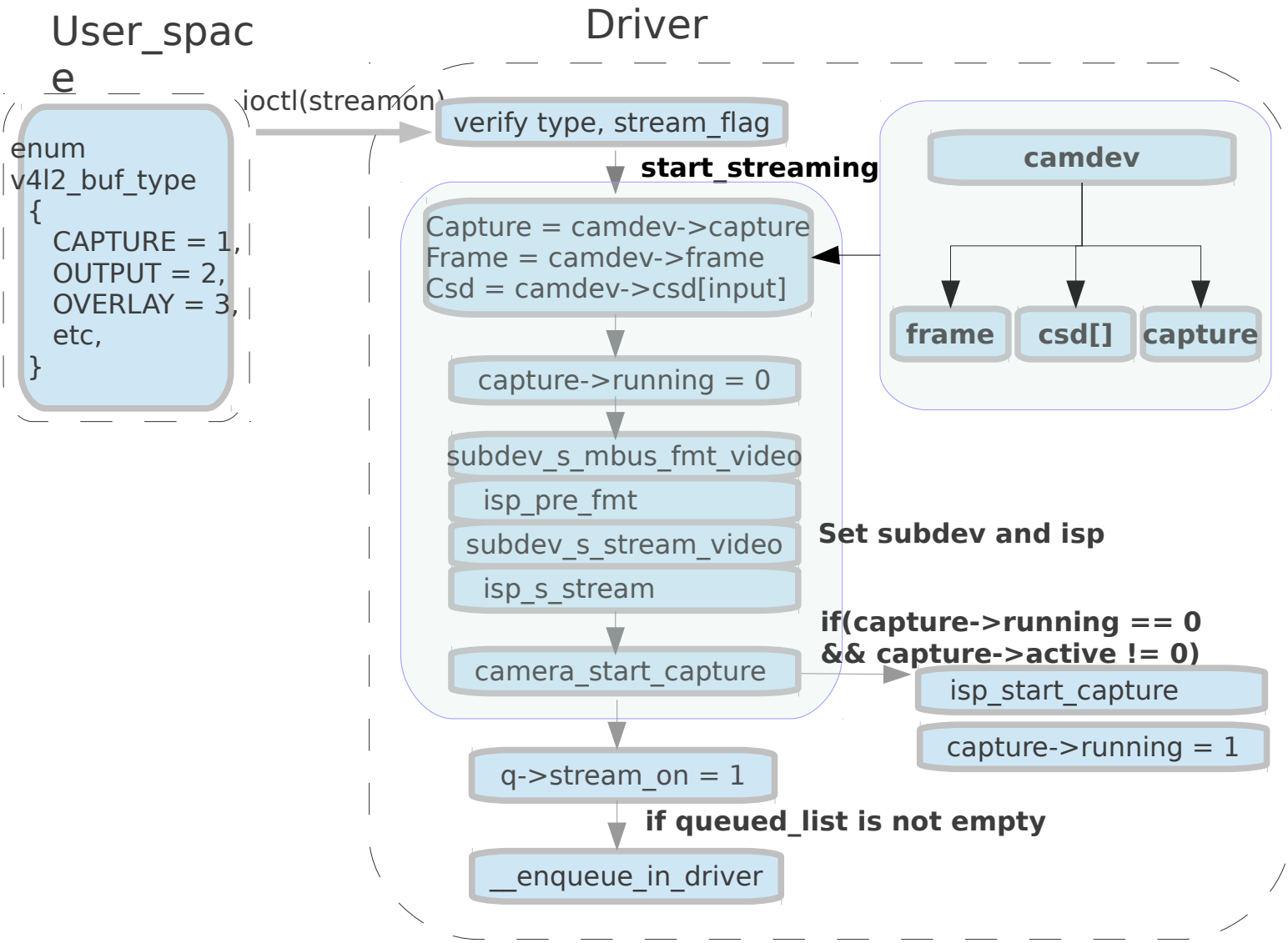
**ovisp_vidioc_streamon**(struct file *file,void *priv, enum v4l2_buf_type i)

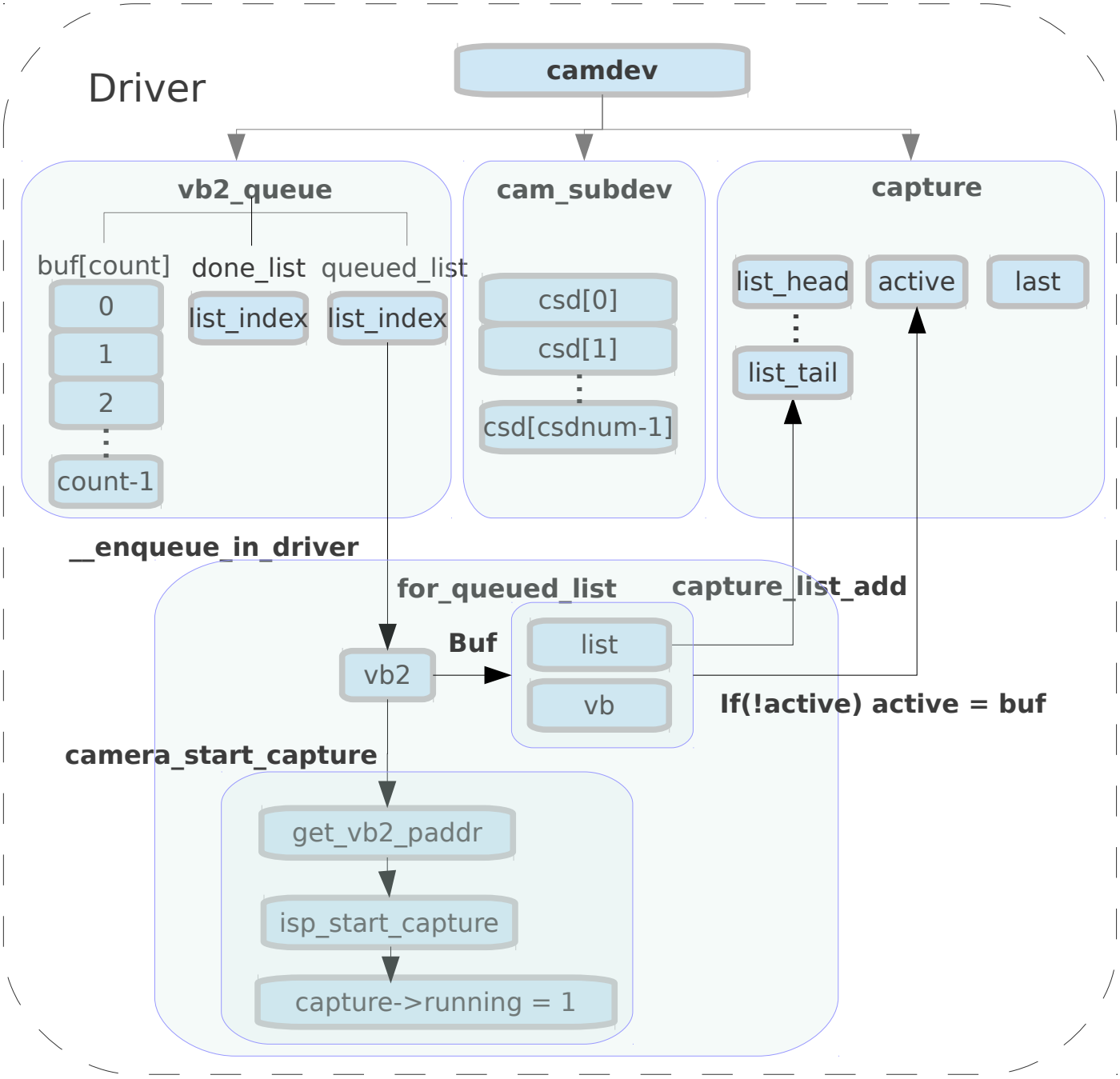
para:

function:
1) verifies current state
2) starts streaming and passes any previously queued buffers to the driver

process:

ovisp_camera_start_streaming(struct ovisp_camera_dev *camdev)




Driver
camdev
vb2_queue
cam_subdev
capture
buf[count]
done_list
queued_list
0
1
2
count-1
list_index
list_index
csd[0]
csd[1]
csd[csdnum-1]
list_head
active
last
list_tail
__enqueue_in_driver
for_queued_list
capture_list_add
Buf
vb2
list
vb
If(!active) active = buf
camera_start_capture
get_vb2_paddr
isp_start_capture
capture->running = 1

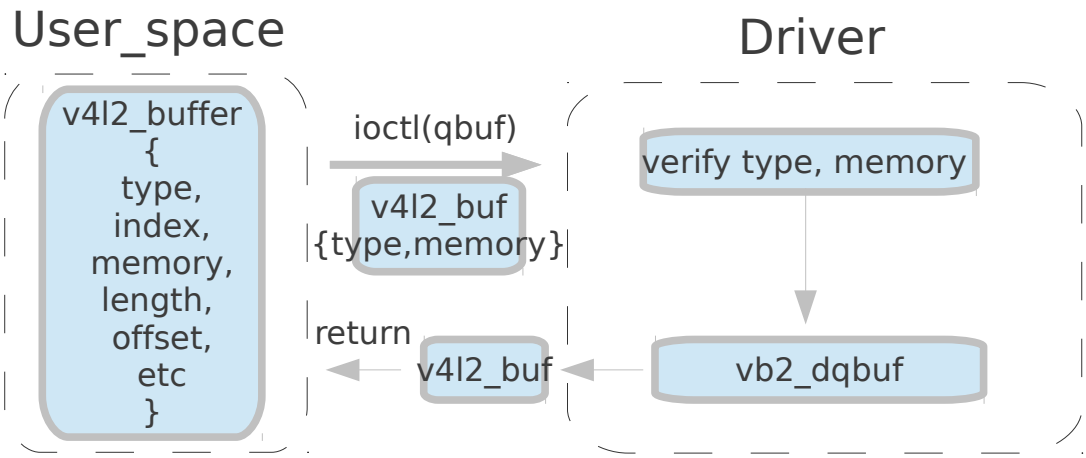
**ovisp_vidioc_dqbuf**(struct file *file, void *priv, struct v4l2_buffer *p)

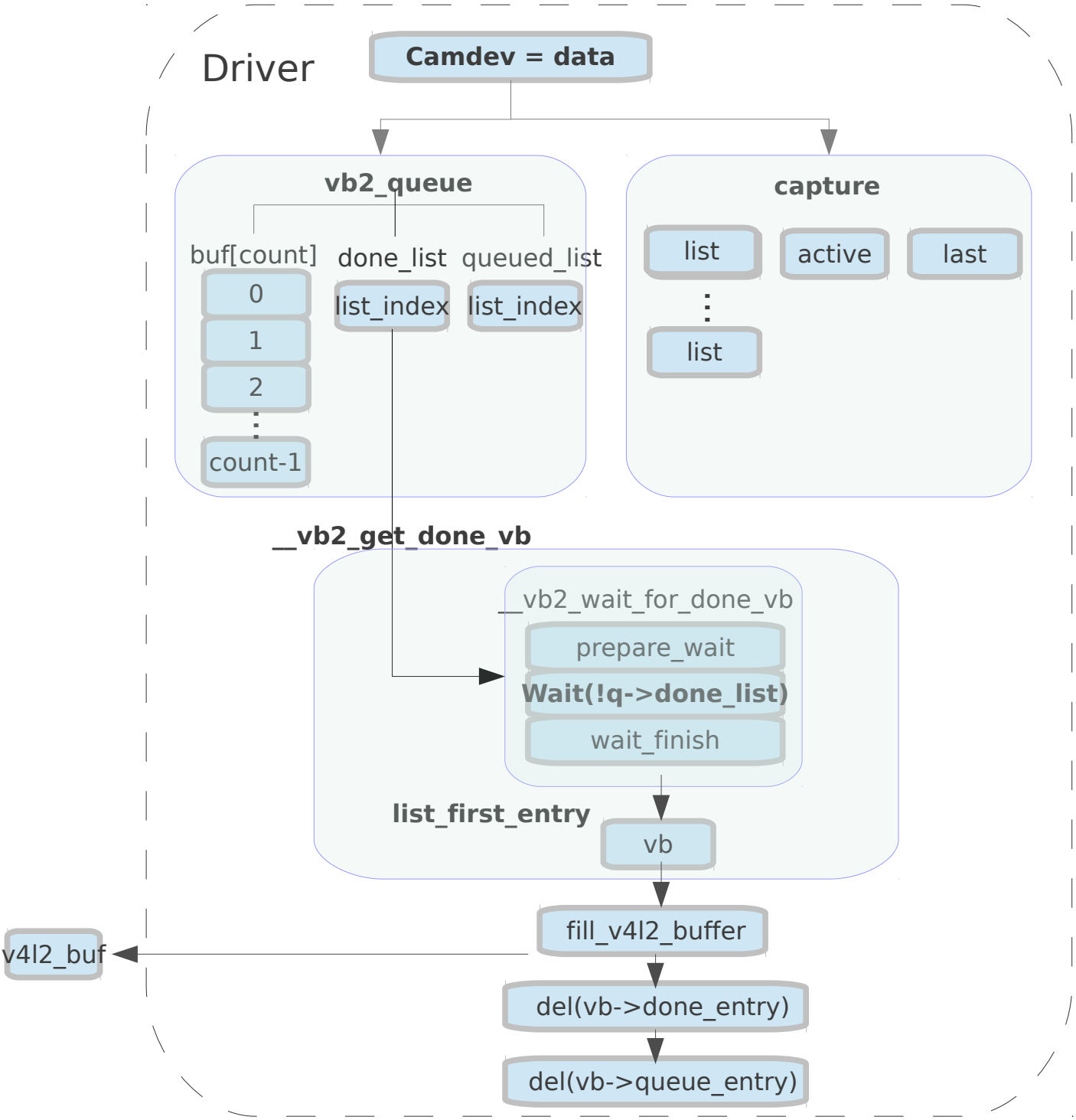
para:

Function:

1) verifies the passed buffer,
2) calls buf_finish callback in the driver (if provided), in which driver can perform any additional operations that may be required before returning the buffer to userspace, such as cache sync,
3) the buffer struct members are filled with relevant information for the userspace.

Process:



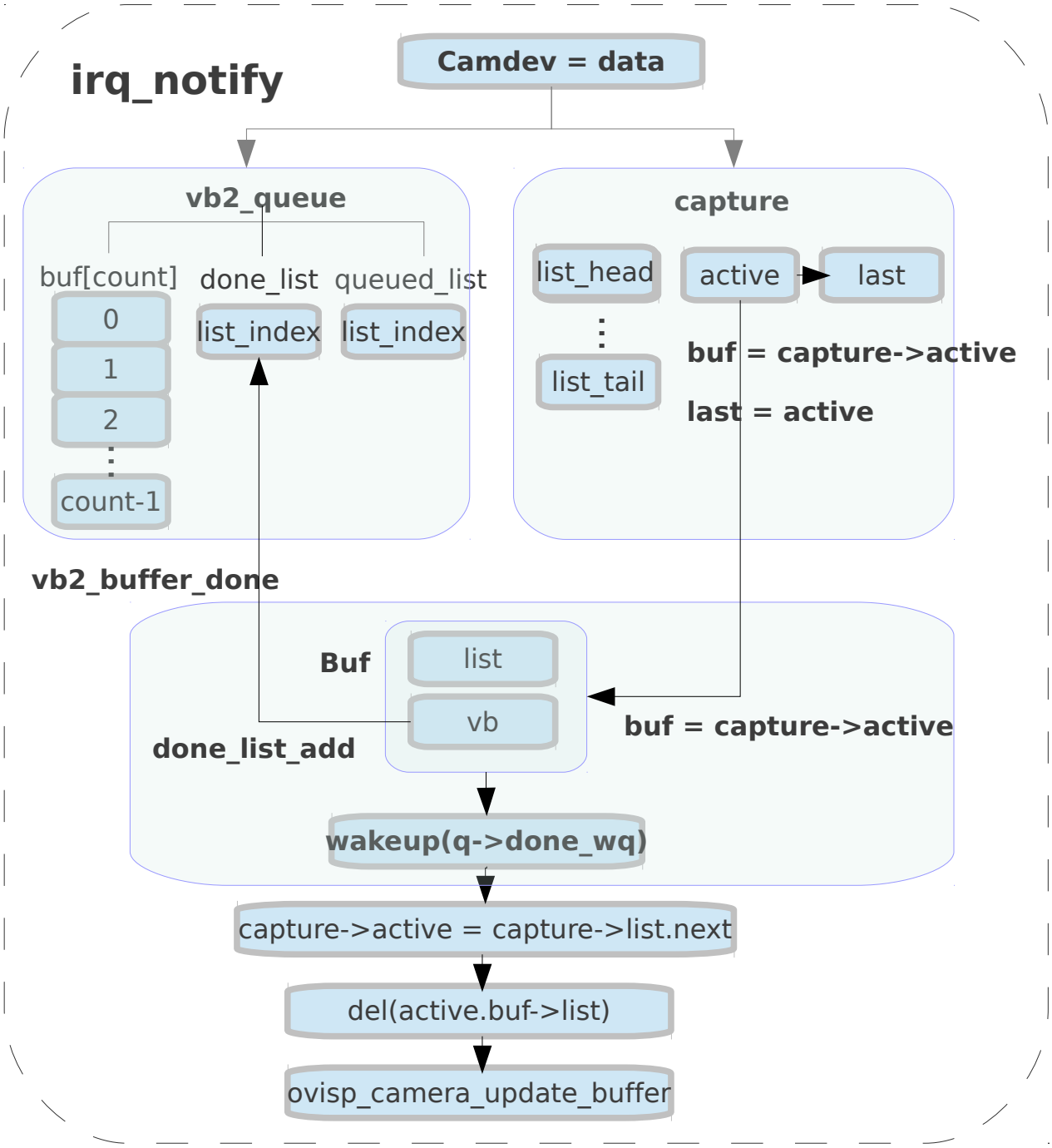

Driver
Camdev = data
vb2_queue
buf[count]
done_list
queued_list
0
1
2
count-1
list_index
list_index
capture
list
active
last
list
__vb2_get_done_vb
__vb2_wait_for_done_vb
prepare_wait
Wait(!q->done_list)
wait_finish
list_first_entry
vb
fill_v4l2_buffer
v4l2_buf
del(vb->done_entry)
del(vb->queue_entry)

**ovisp_camera_irq_notify**(unsigned int status, void *data)

para:

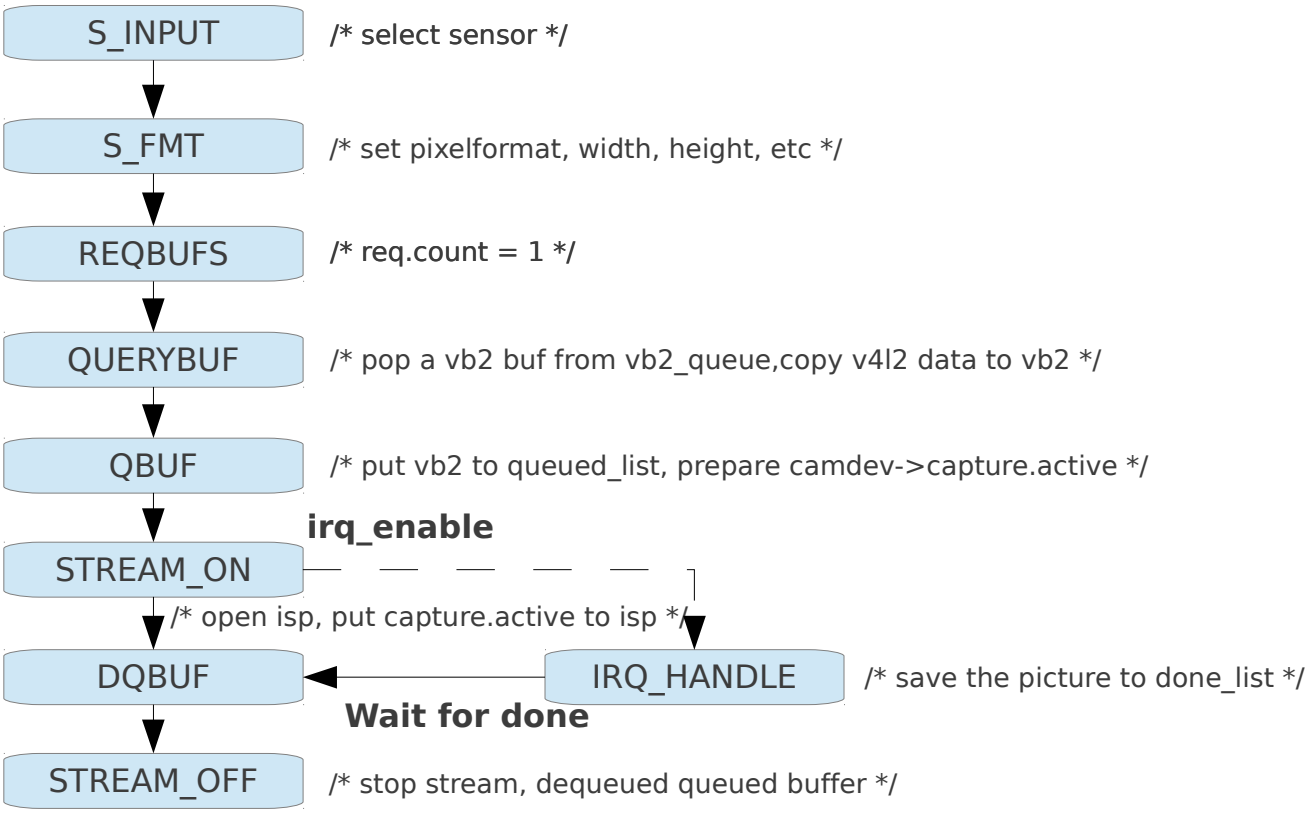
function:
irq function called after hardware operation, capture frame and update the vb2_queue list

process:

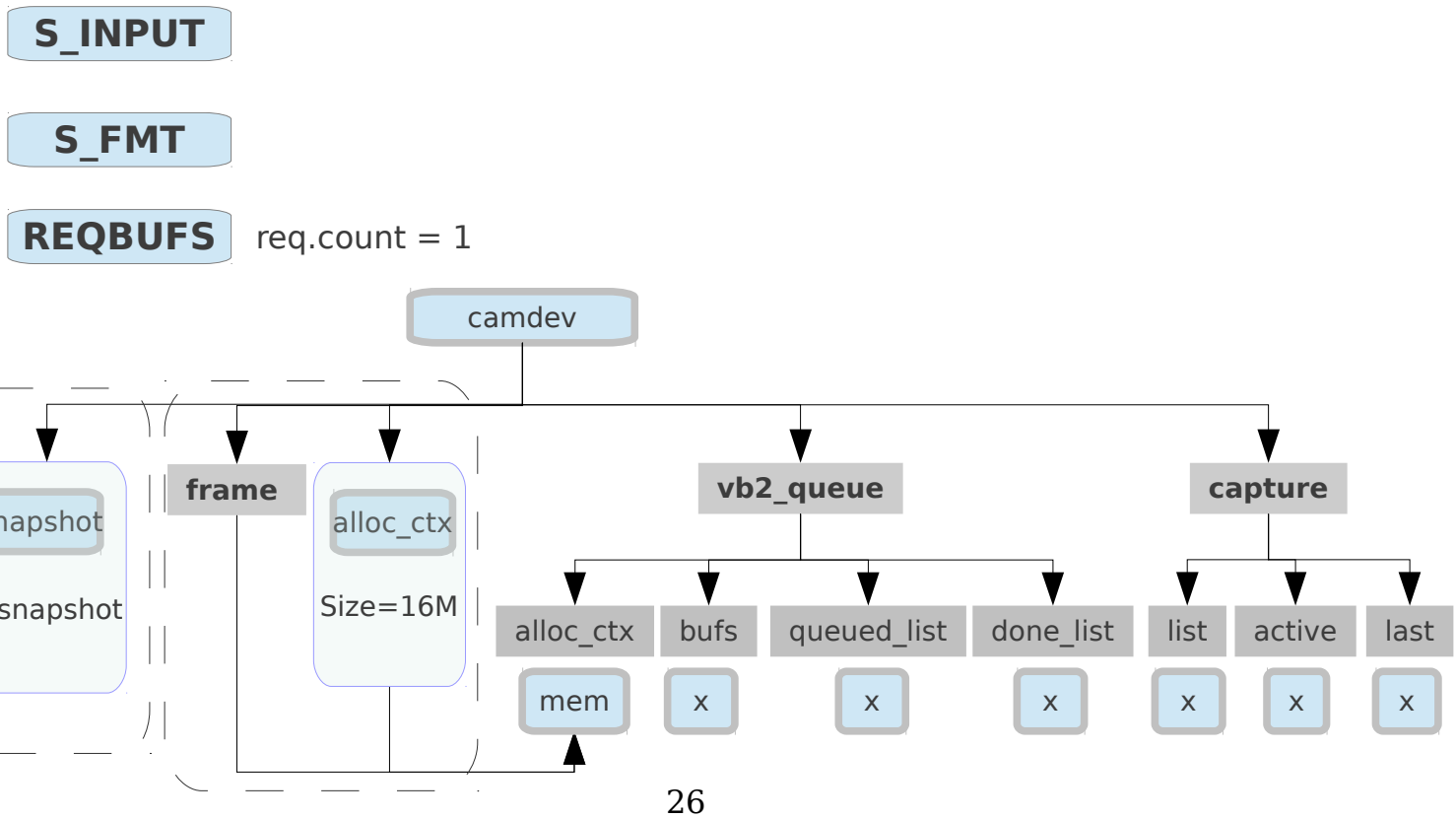


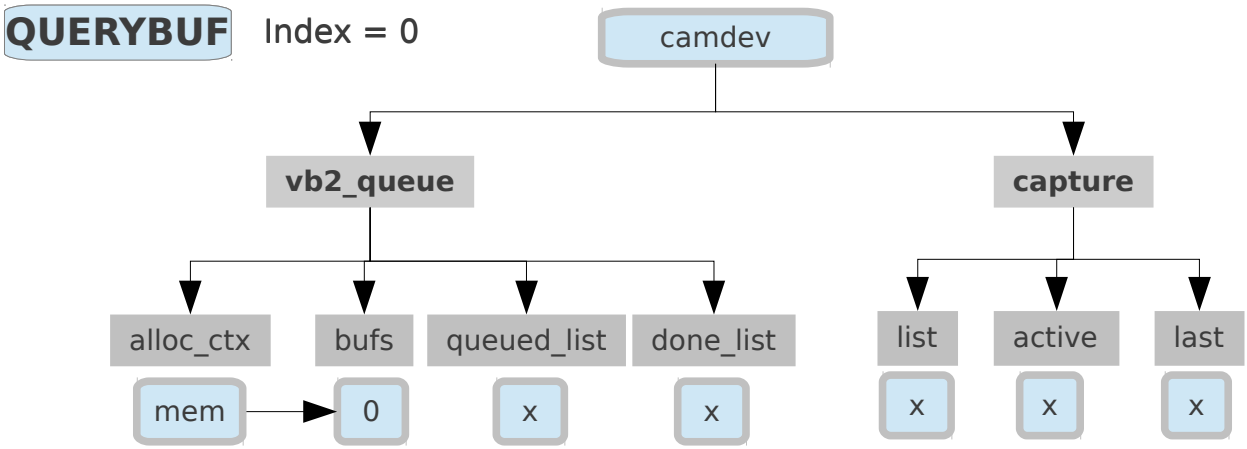
## (6) Main Operations

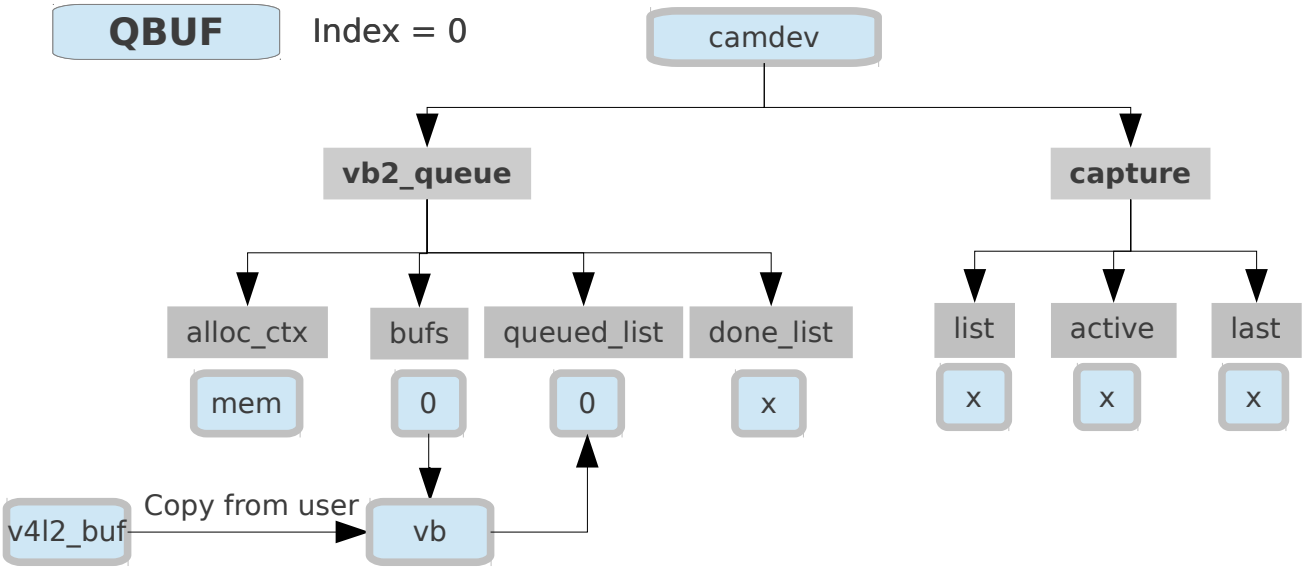
### Capture

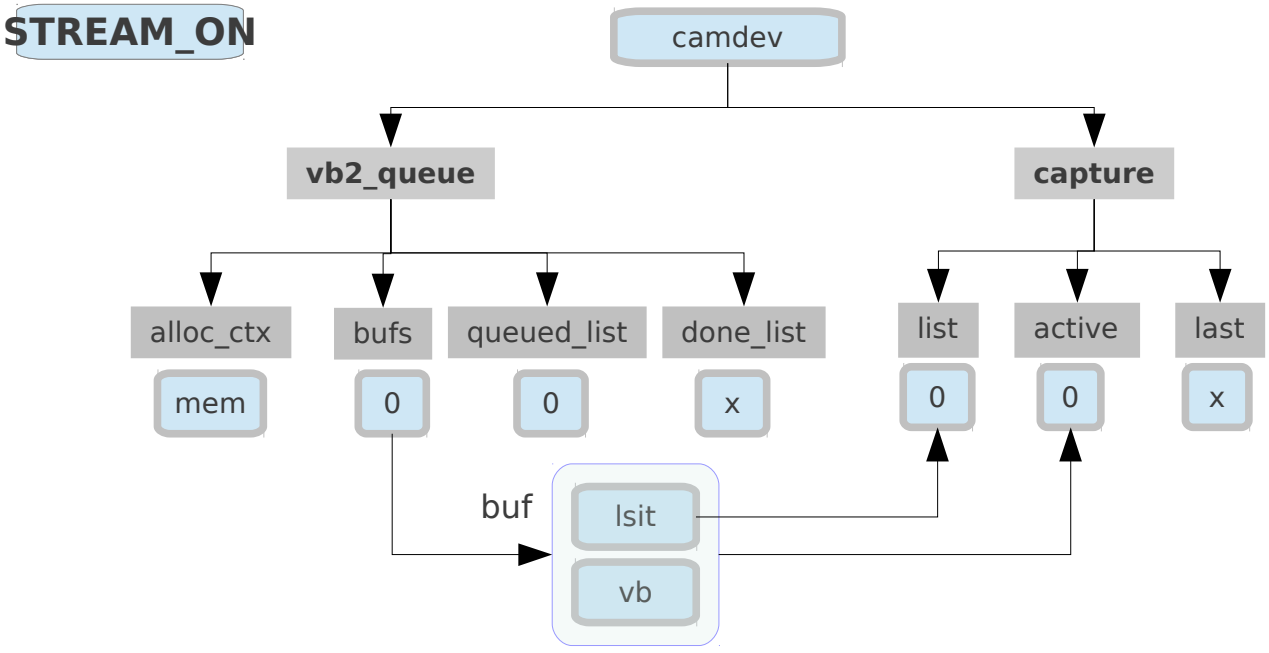


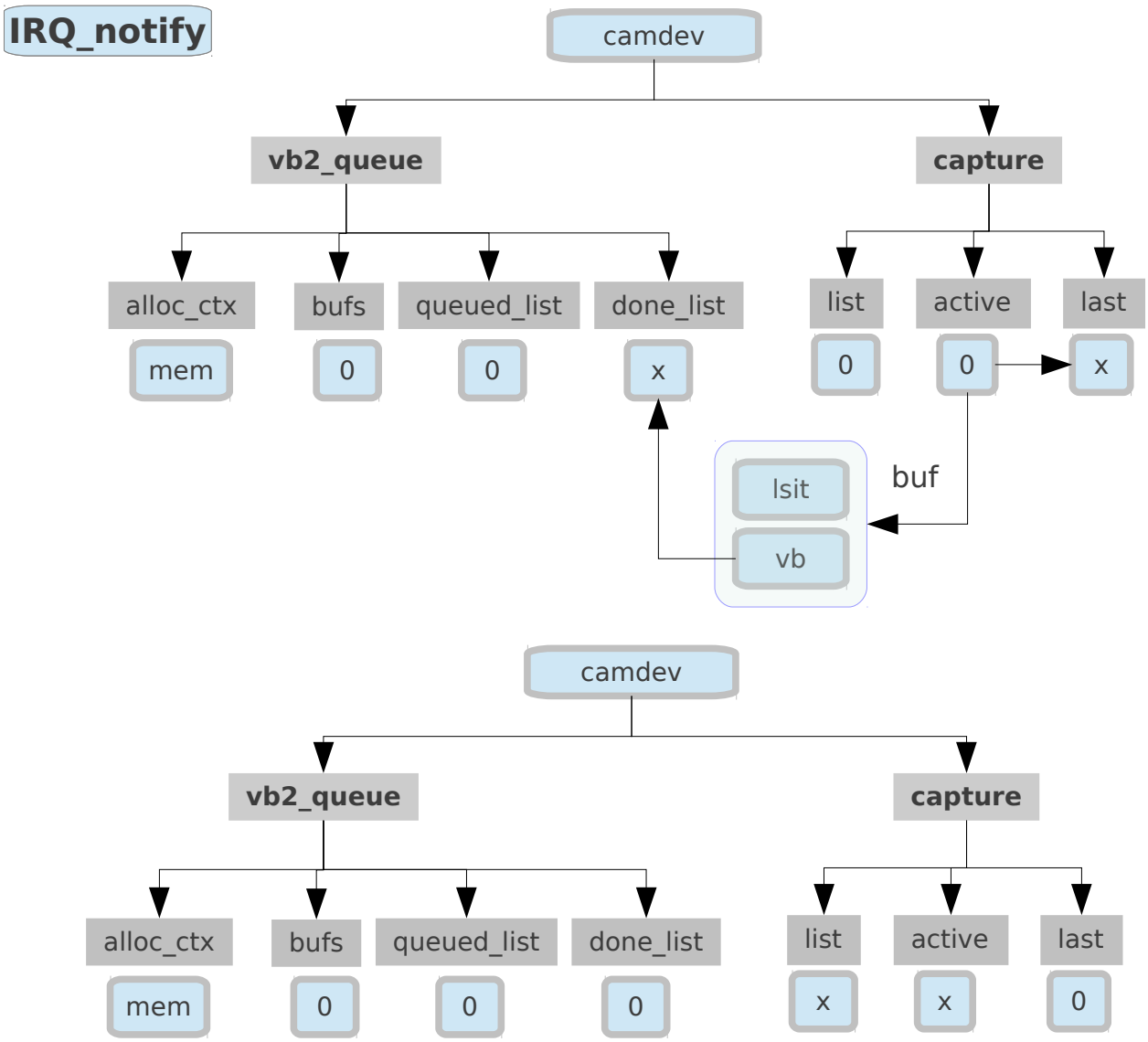

**/* camdev in capture process */**

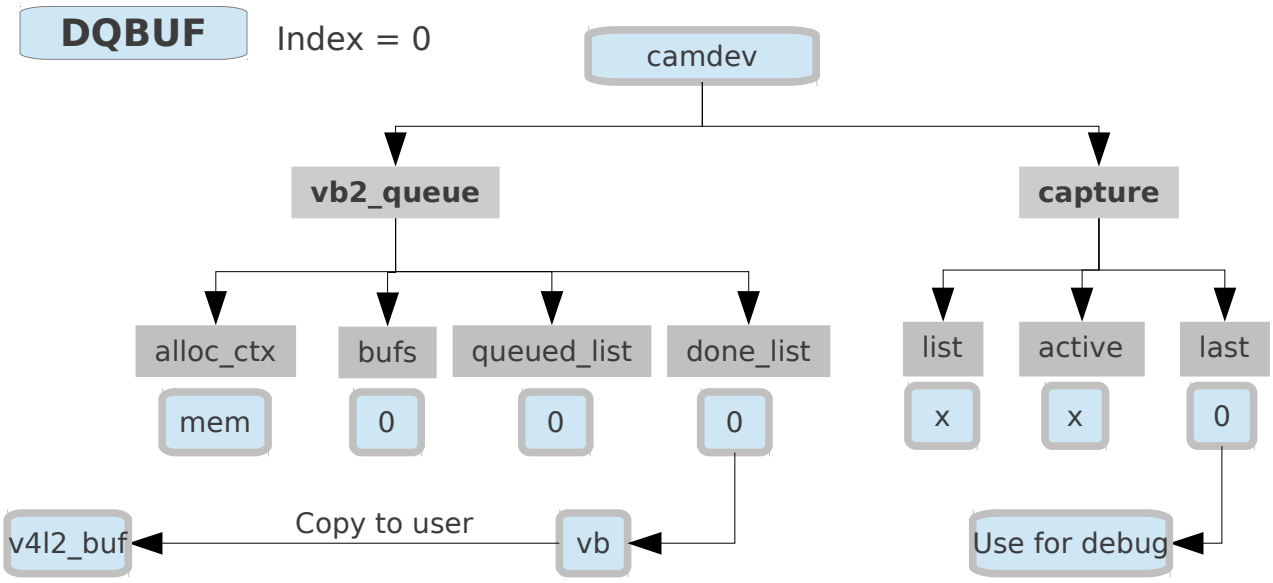


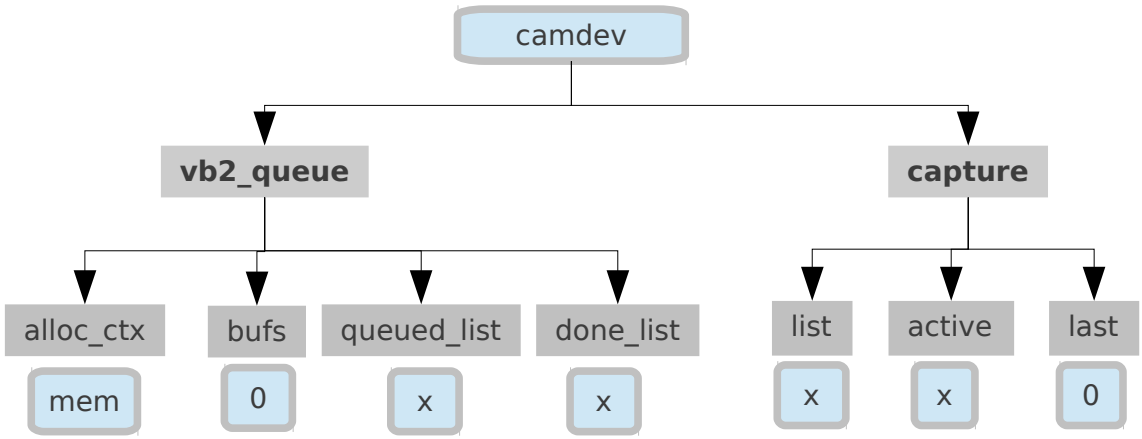

QUERYBUF
Index = 0
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
0
x
x
x
x
x

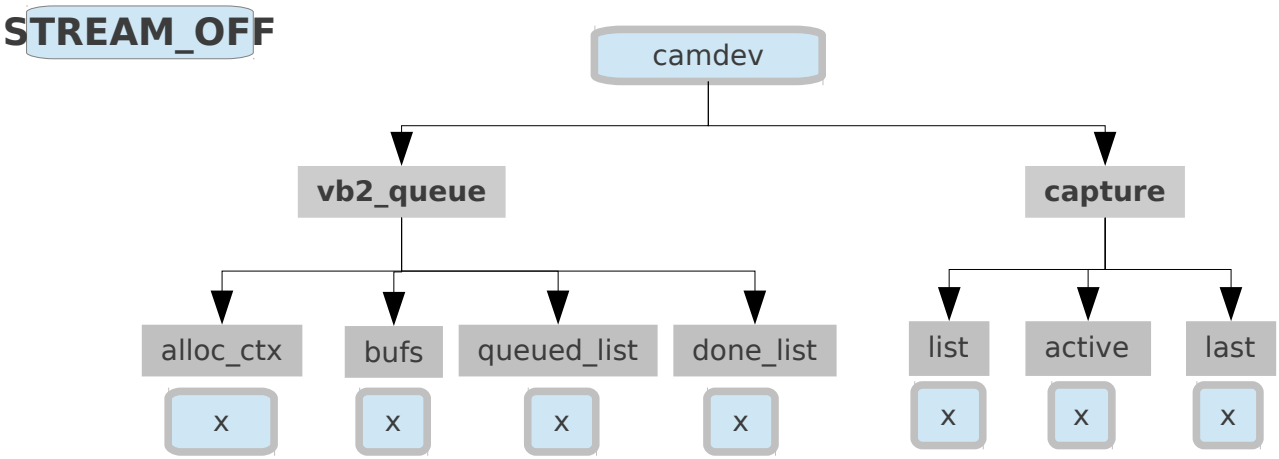

QBUF
Index = 0
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
0
0
x
x
x
x
v4l2_buf
Copy from user
vb


STREAM_ON
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
0
0
x
0
0
x
buf
lsit
vb

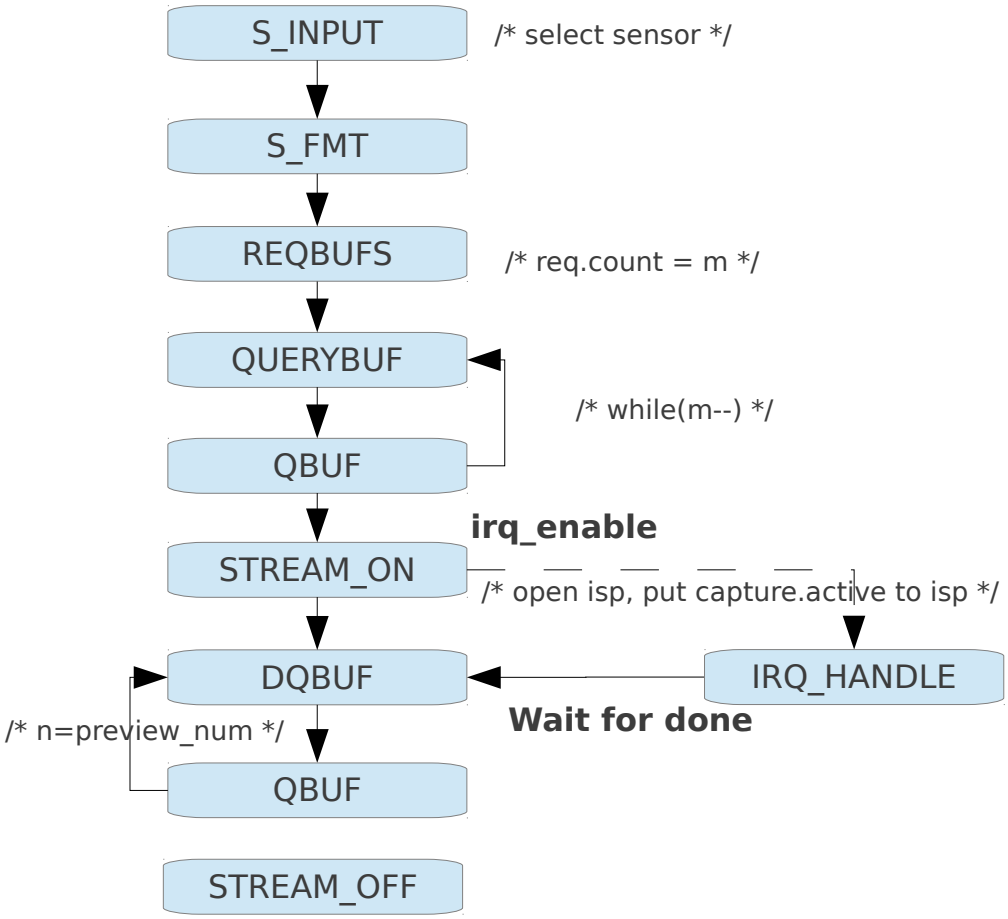

IRQ_notify
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
0
0
x
0
0
x
lsit
vb
buf
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
0
0
0
x
x
0


DQBUF
Index = 0
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
0
0
0
x
x
0
v4l2_buf
Copy to user
vb
Use for debug


camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
0
x
x
x
x
0


STREAM_OFF
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
x
x
x
x
x
x
x

## Preview

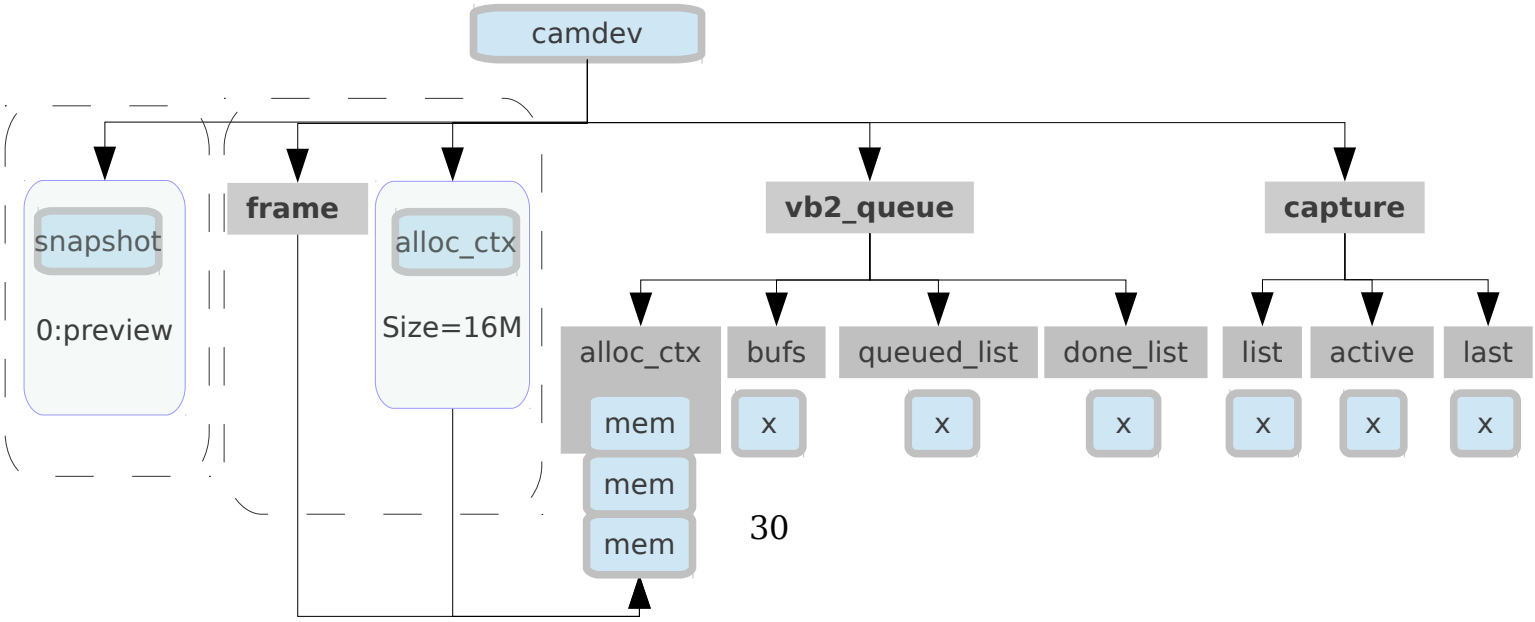


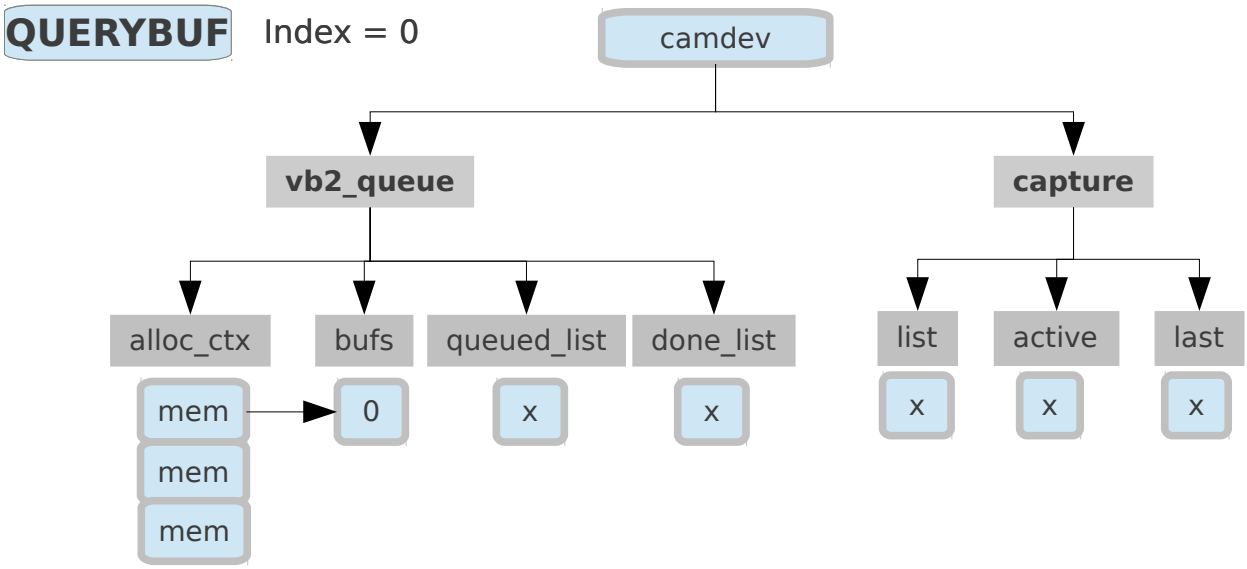

**/* camdev in capture process */**

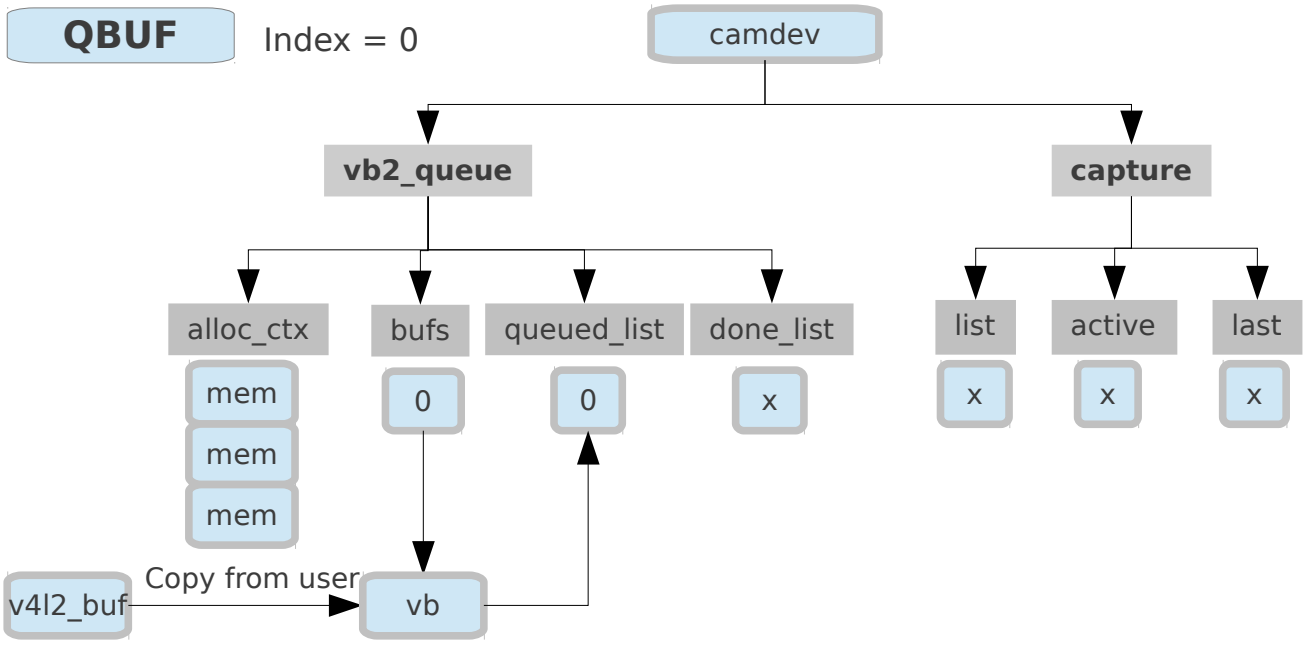
**S_INPUT**

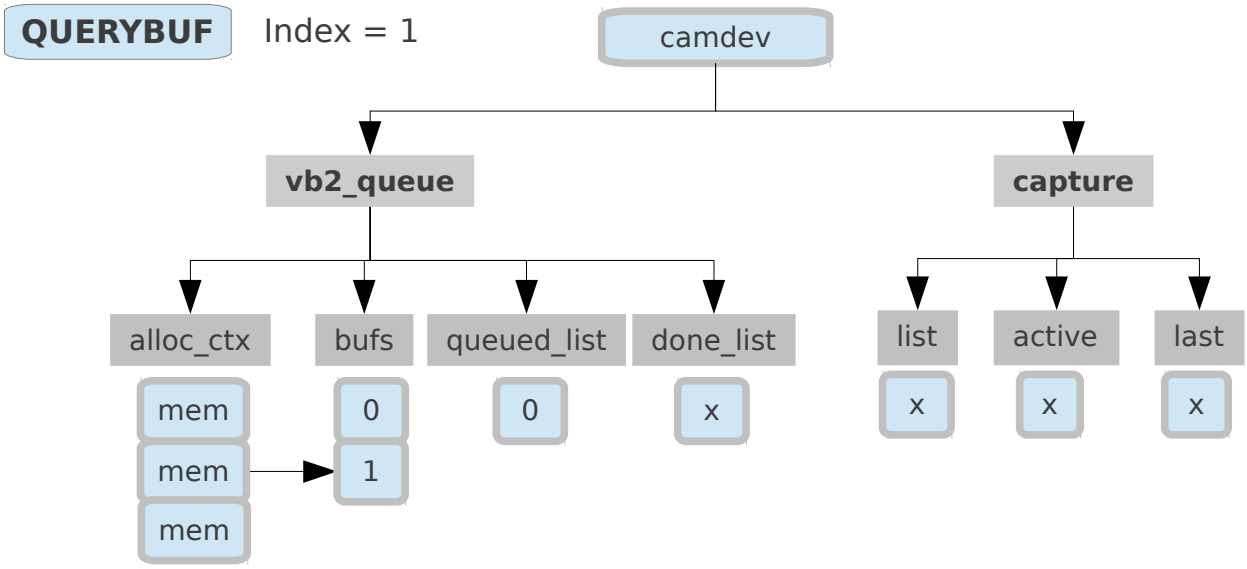
**S_FMT**

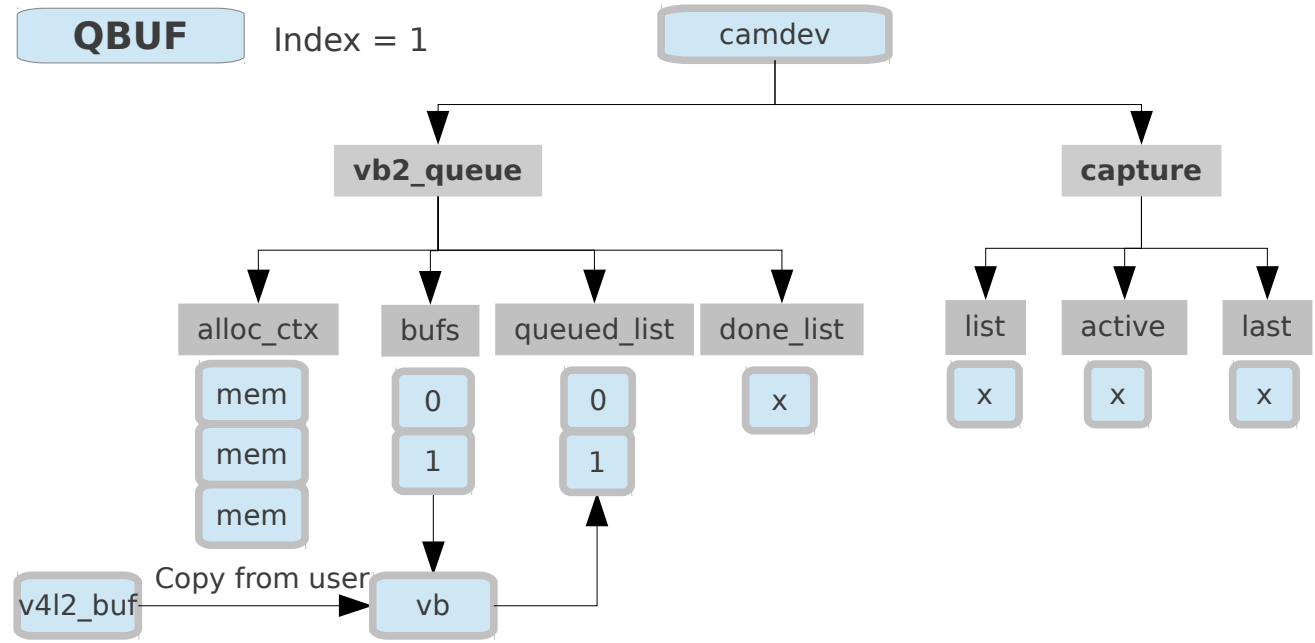
**REQBUFS** req.count = 3

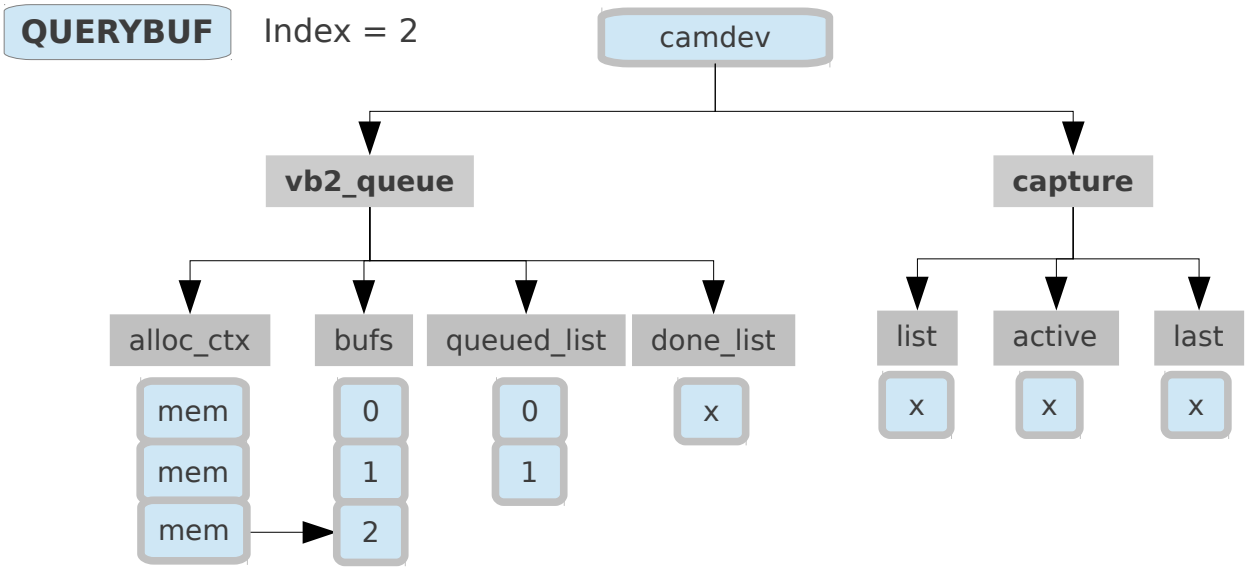


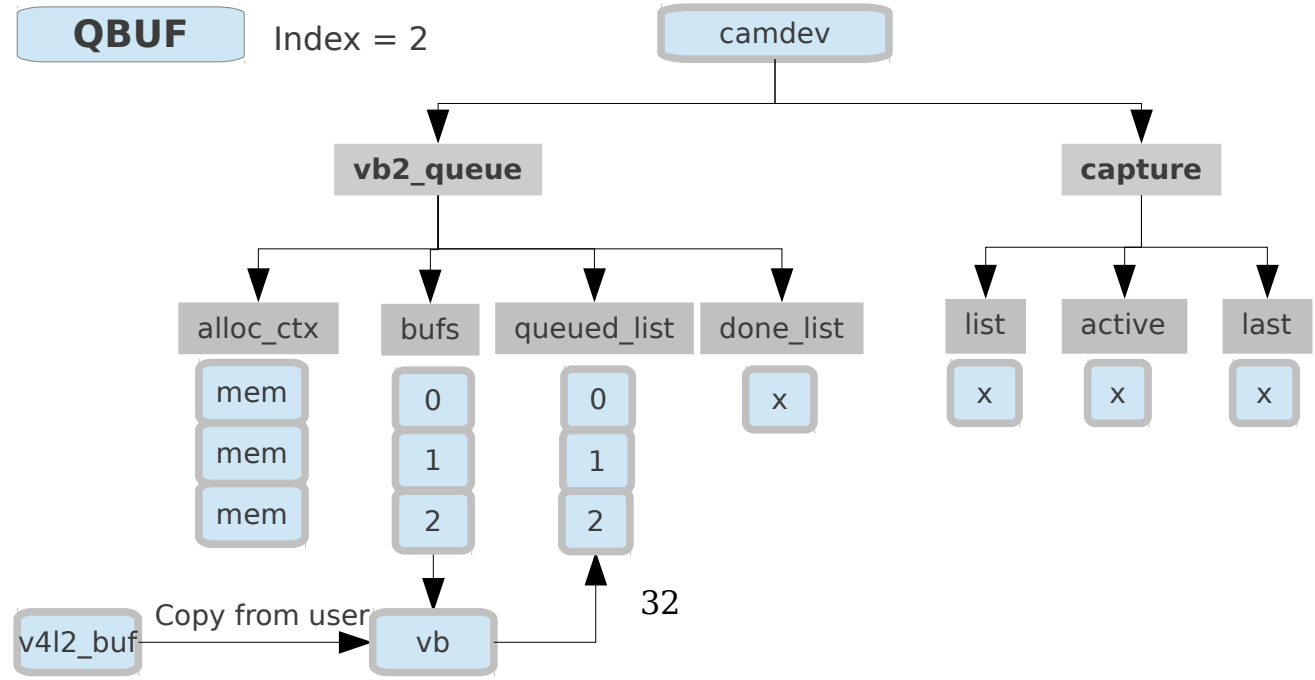

QUERYBUF
Index = 0
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
x
x
x
x
x

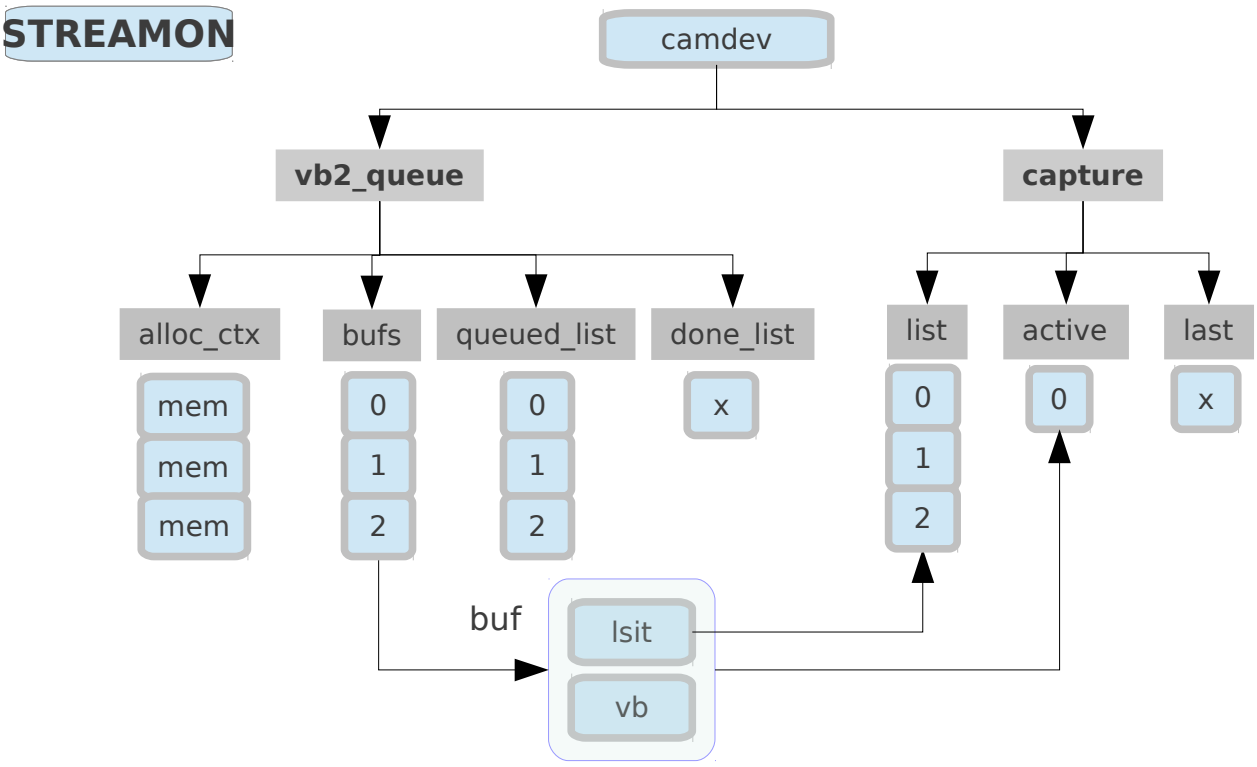

QBUF
Index = 0
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
0
x
x
x
x
v4l2_buf
Copy from user
vb

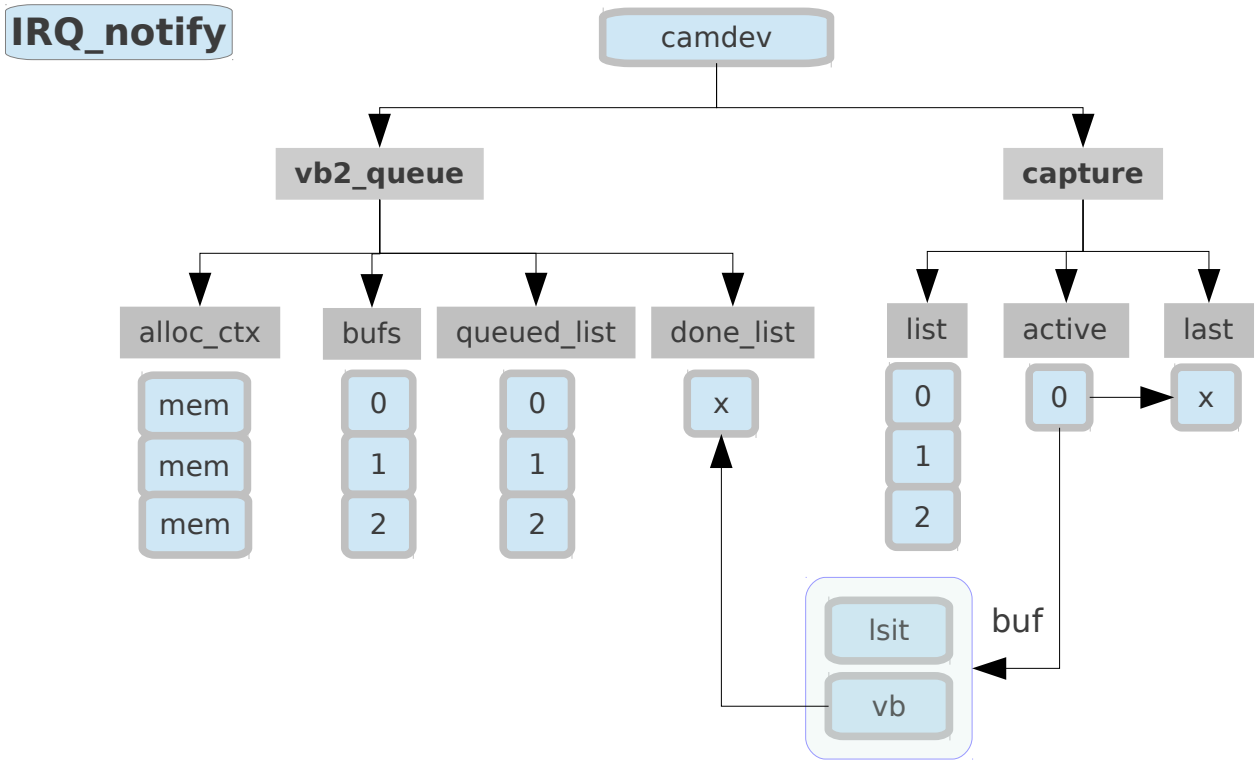

QUERYBUF
Index = 1
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
0
x
x
x
x

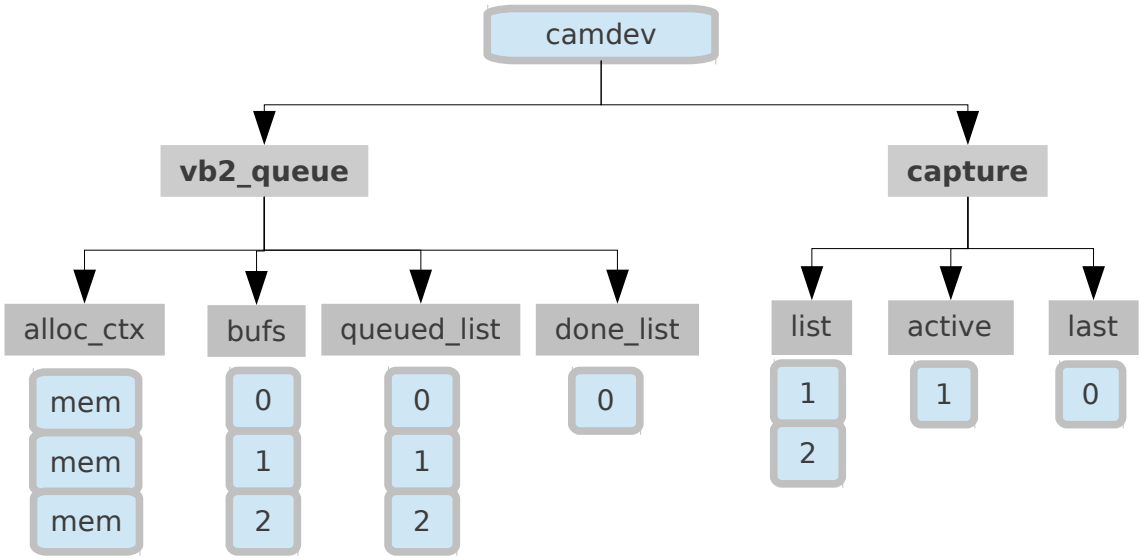

QBUF
Index = 1
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
0
1
x
x
x
x
Copy from user
v4l2_buf
vb


QUERYBUF
Index = 2
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
2
0
1
x
x
x
x

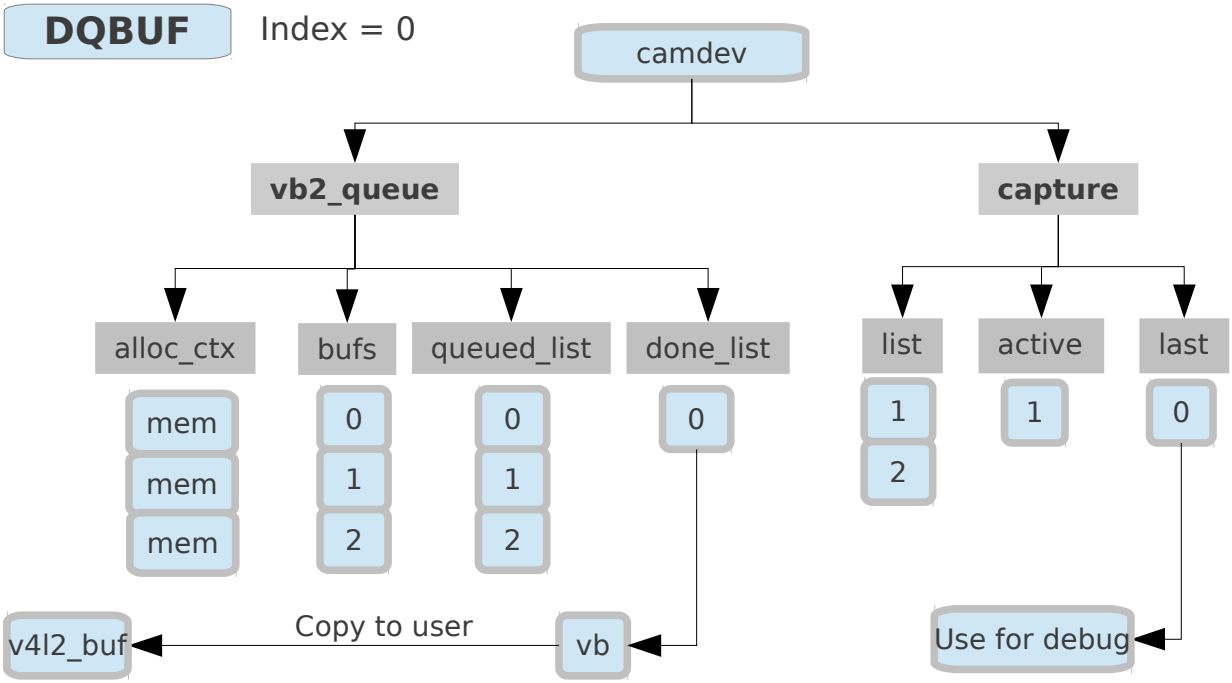

QBUF
Index = 2
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
2
0
1
2
x
x
x
x
Copy from user
v4l2_buf
vb

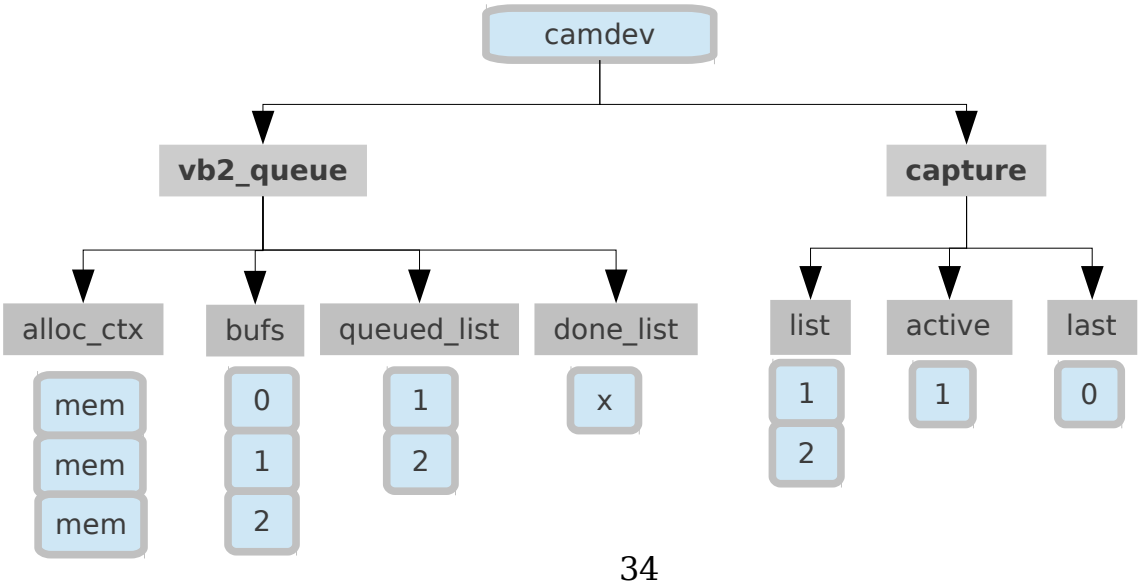

STREAMON
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
2
0
1
2
x
0
1
2
0
x
buf
lsit
vb

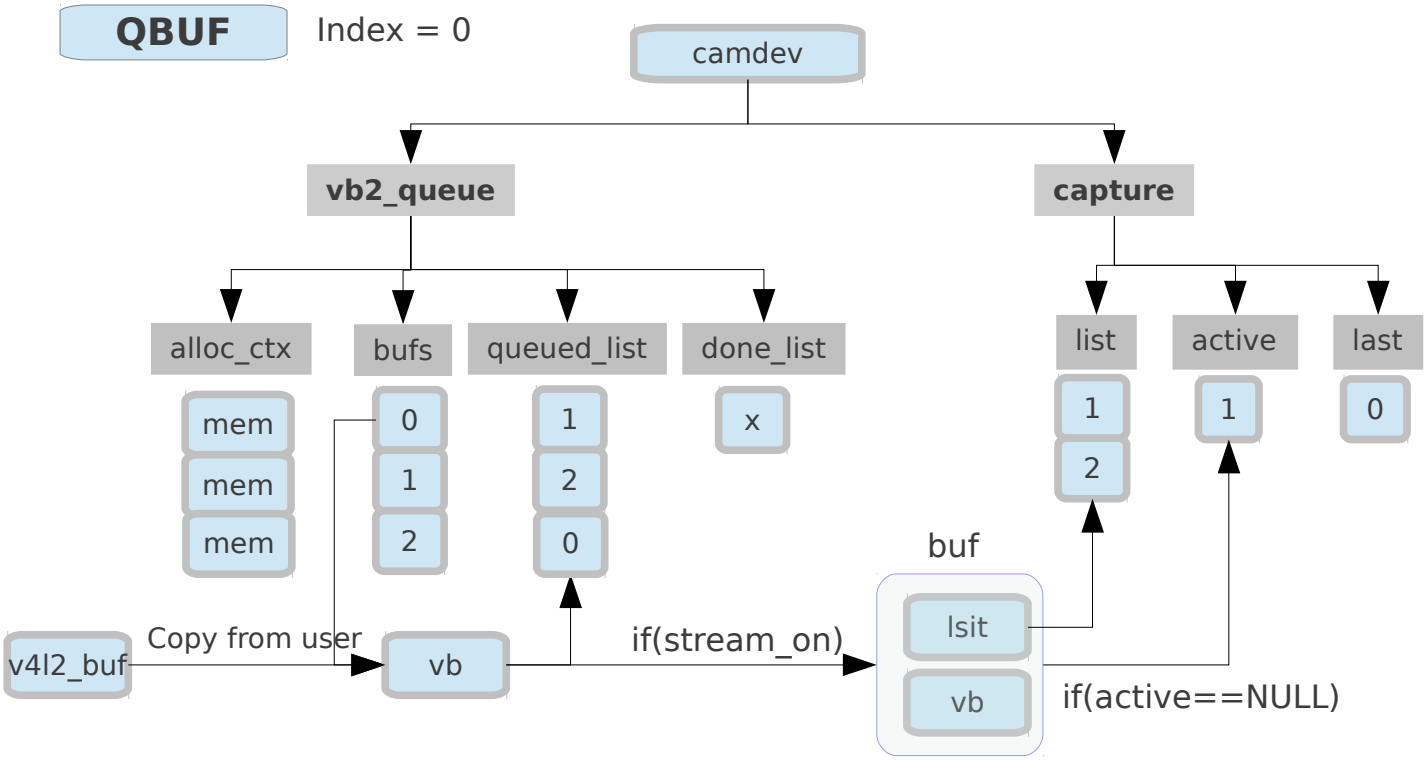

IRQ_notify
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
2
0
1
2
x
0
1
2
0
x
lsit
vb
buf

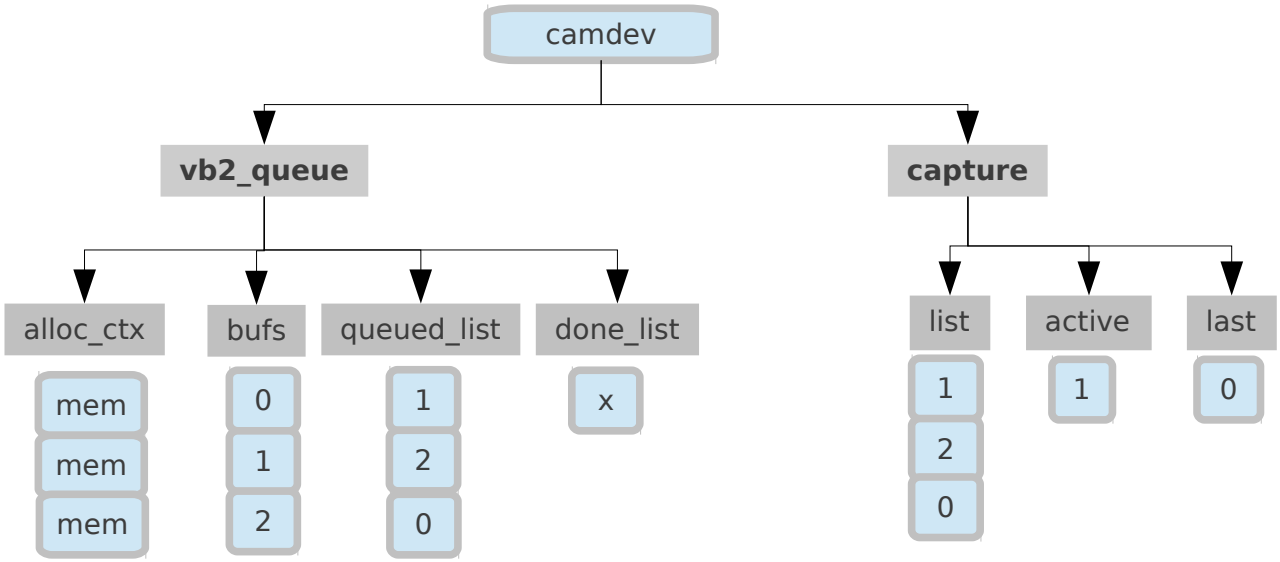



**DQBUF** Index = 0



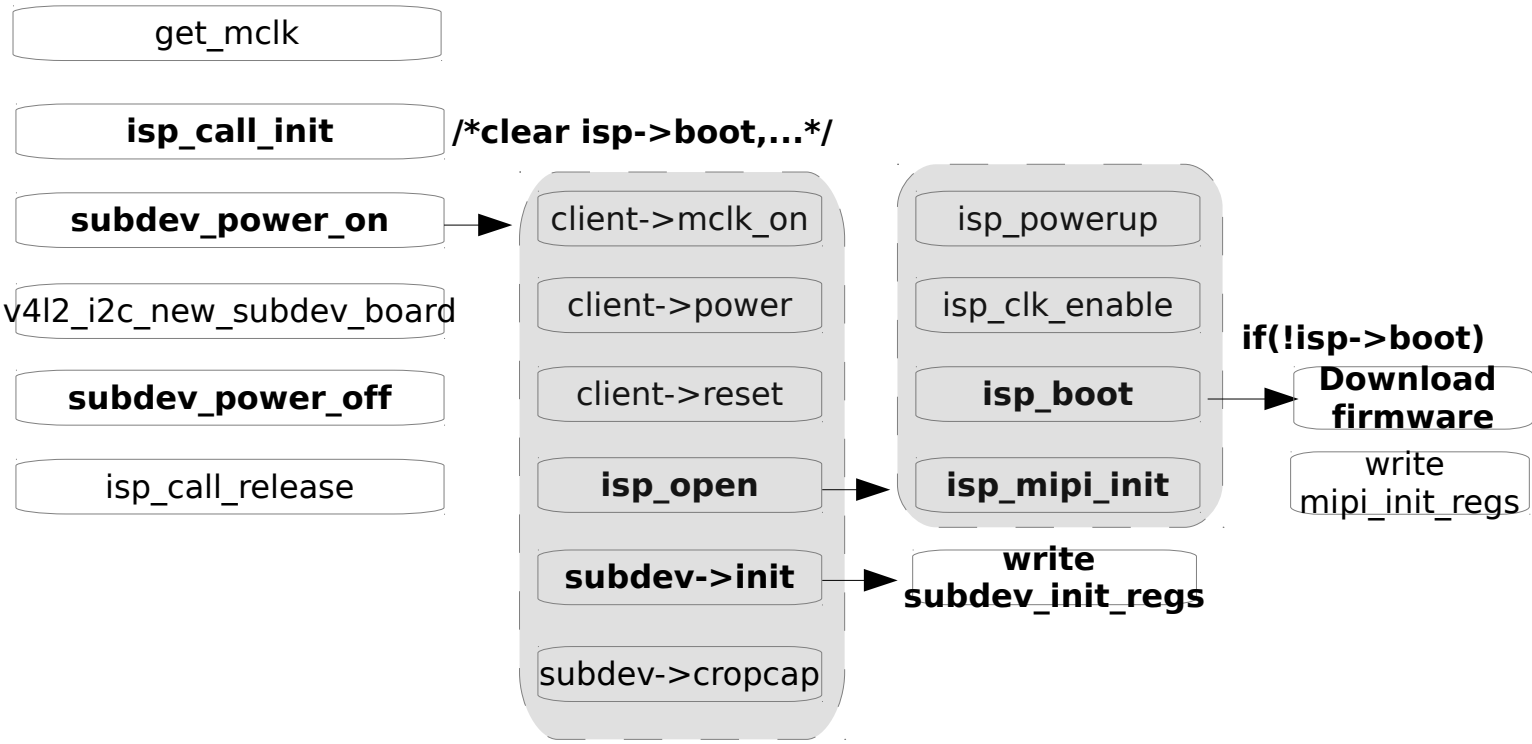




QBUF
Index = 0
camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
2
1
2
0
x
1
2
1
0
buf
lsit
vb
v4l2_buf
Copy from user
vb
if(stream_on)
if(active==NULL)

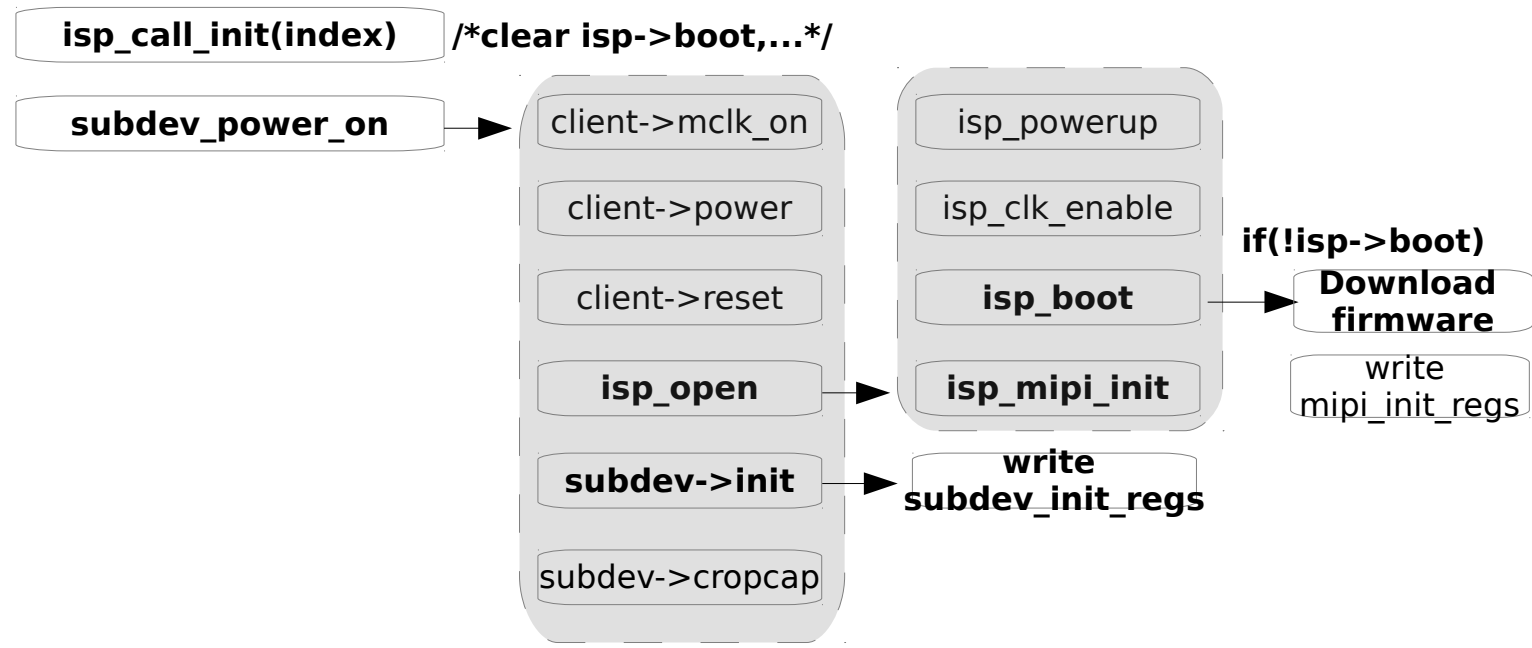

camdev
vb2_queue
capture
alloc_ctx
bufs
queued_list
done_list
list
active
last
mem
mem
mem
0
1
2
1
2
0
x
1
2
0
1
0

## (7)ISP CONFIG

### camera_init_client




### s_input



## s_fmt

## stream_on

start_streaming