static void vi5\_capture\_dequeue(struct tegra\_channel \*chan,

 struct tegra\_channel\_buffer \*buf)

{

 int err = 0;

 unsigned long flags;

 struct tegra\_mc\_vi \*vi = chan->vi;

 struct vb2\_v4l2\_buffer \*vb = &buf->buf;

 struct timespec ts;

 struct capture\_descriptor \*descr =

 &chan->request[buf->capture\_descr\_index];

 if (buf->vb2\_state != VB2\_BUF\_STATE\_ACTIVE)

 goto rel\_buf;

 /\* Dequeue a frame and check its capture status \*/

 err = vi\_capture\_status(chan->tegra\_vi\_channel, CAPTURE\_TIMEOUT\_MS);

 descr->status.status = CAPTURE\_STATUS\_SUCCESS;// @@@@ added 2-24-2021

 printk(“[%s][%s][%d][descr->status.status:%x]\n”, “marcus”, \_\_func\_\_, \_\_LINE\_\_, descr->status.status);// // @@@@ added 2-24-2021

 if (err) {

 if (err == -ETIMEDOUT) {

 dev\_err(vi->dev,

 “uncorr\_err: request timed out after %d ms\n”,

 CAPTURE\_TIMEOUT\_MS);

 } else {

 dev\_err(vi->dev, “uncorr\_err: request err %d\n”, err);

 }

 goto uncorr\_err;

 } else if (descr->status.status != CAPTURE\_STATUS\_SUCCESS) {

 if ((descr->status.flags

 & CAPTURE\_STATUS\_FLAG\_CHANNEL\_IN\_ERROR) != 0) {

 chan->queue\_error = true;

 dev\_err(vi->dev, “uncorr\_err: flags %d, err\_data %d\n”,

 descr->status.flags, descr->status.err\_data);

 } else {

 dev\_warn(vi->dev,

 “corr\_err: discarding frame %d, flags: %d, ”

 “err\_data %d\n”,

 descr->status.frame\_id, descr->status.flags,

 descr->status.err\_data);

 buf->vb2\_state = VB2\_BUF\_STATE\_REQUEUEING;

 goto done;

 }

 }

 buf->vb2\_state = VB2\_BUF\_STATE\_DONE;

 /\* Read SOF from capture descriptor \*/

 ts = ns\_to\_timespec((s64)descr->status.sof\_timestamp);

 trace\_tegra\_channel\_capture\_frame(“sof”, ts);

#if LINUX\_VERSION\_CODE < KERNEL\_VERSION(4, 9, 0)

 /\* update time stamp of the buffer \*/

 vb->timestamp.tv\_sec = ts.tv\_sec;

 vb->timestamp.tv\_usec = ts.tv\_nsec / NSEC\_PER\_USEC;

#else

 vb->vb2\_buf.timestamp = descr->status.sof\_timestamp;

#endif

 /\* Read EOF from capture descriptor \*/

 ts = ns\_to\_timespec((s64)descr->status.eof\_timestamp);

 trace\_tegra\_channel\_capture\_frame(“eof”, ts);

done:

 spin\_lock\_irqsave(&chan->capture\_state\_lock, flags);

 if (chan->capture\_state != CAPTURE\_ERROR) {

 chan->capture\_reqs\_enqueued -= 1;

 chan->capture\_state = CAPTURE\_GOOD;

 }

 spin\_unlock\_irqrestore(&chan->capture\_state\_lock, flags);

 wake\_up\_interruptible(&chan->start\_wait);

 goto rel\_buf;

uncorr\_err:

 spin\_lock\_irqsave(&chan->capture\_state\_lock, flags);

 chan->capture\_state = CAPTURE\_ERROR;

 spin\_unlock\_irqrestore(&chan->capture\_state\_lock, flags);

 buf->vb2\_state = VB2\_BUF\_STATE\_ERROR;

rel\_buf:

 vi5\_release\_buffer(chan, buf);

}