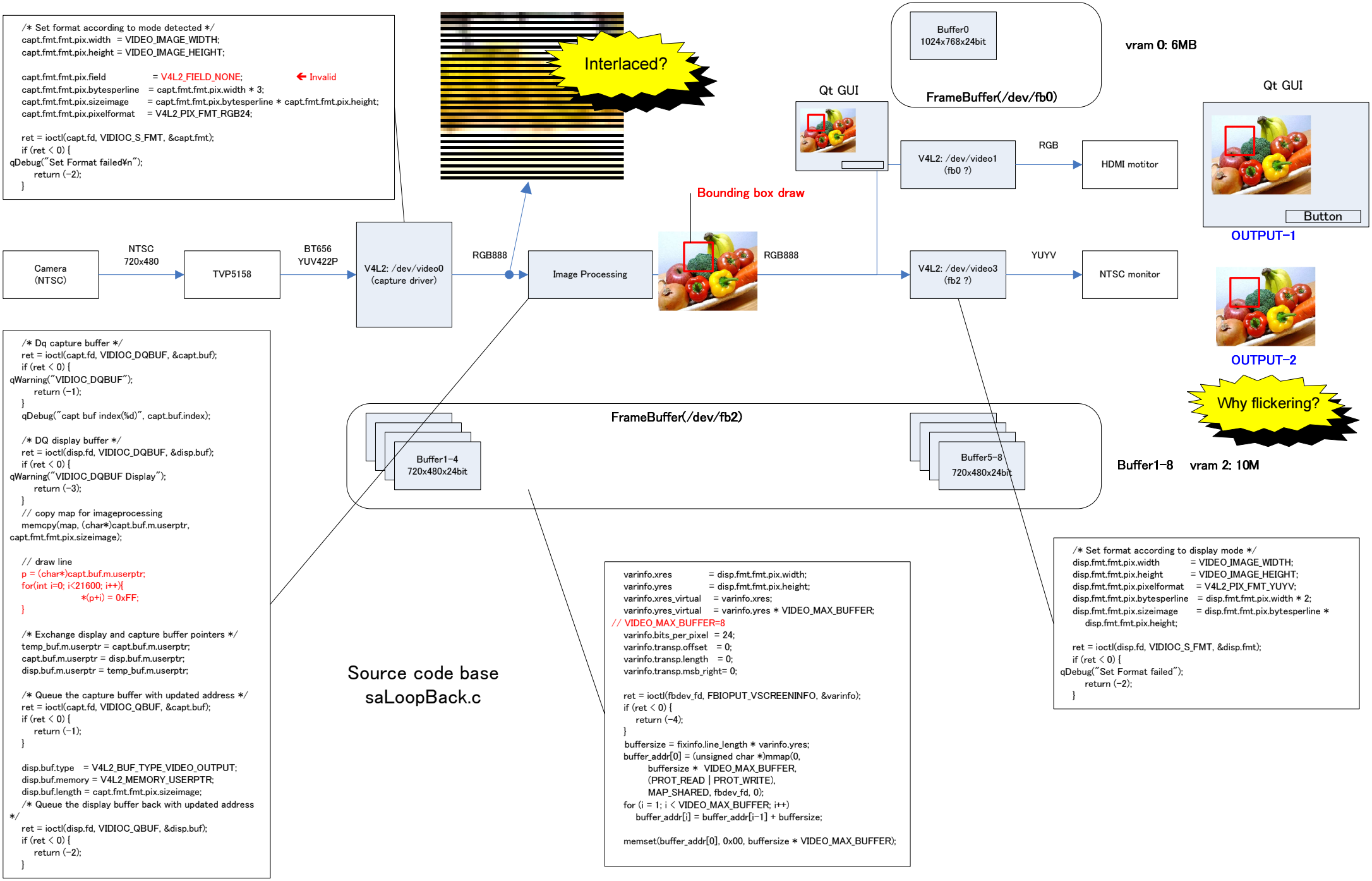
/* Set format according to mode detected */
capt.fmt.fmt.pix.width = VIDEO_IMAGE_WIDTH;
capt.fmt.fmt.pix.height = VIDEO_IMAGE_HEIGHT;
capt.fmt.fmt.pix.field = V4L2_FIELD_NONE;
Invalid
capt.fmt.fmt.pix.bytesperline = capt.fmt.fmt.pix.width * 3;
capt.fmt.fmt.pix.sizeimage = capt.fmt.fmt.pix.bytesperline * capt.fmt.fmt.pix.height;
capt.fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_RGB24;
ret = ioctl(capt.fd, VIDIOC_S_FMT, &capt.fmt);
if (ret < 0) {
qDebug("Set Format failed¥n");
return (-2);
}
Interlaced?
Buffer0
1024x768x24bit
vram 0: 6MB
FrameBuffer(/dev/fb0)
Qt GUI
V4L2: /dev/video1
(fb0 ?)
RGB
HDMI motitor
Qt GUI
Button
OUTPUT-1
Bounding box draw
Camera
(NTSC)
NTSC
720x480
TVP5158
BT656
YUV422P
V4L2: /dev/video0
(capture driver)
RGB888
Image Processing
RGB888
V4L2: /dev/video3
(fb2 ?)
YUYV
NTSC monitor
OUTPUT-2
Why flickering?
FrameBuffer(/dev/fb2)
Buffer1-4
720x480x24bit
Buffer5-8
720x480x24bit
Buffer1-8 vram 2: 10M
/* Dq capture buffer */
ret = ioctl(capt.fd, VIDIOC_DQBUF, &capt.buf);
if (ret < 0) {
qWarning("VIDIOC_DQBUF");
return (-1);
}
qDebug("capt buf index(%d)", capt.buf.index);
/* DQ display buffer */
ret = ioctl(disp.fd, VIDIOC_DQBUF, &disp.buf);
if (ret < 0) {
qWarning("VIDIOC_DQBUF Display");
return (-3);
}
// copy map for imageprocessing
memcpy(map, (char*)capt.buf.m.userptr,
capt.fmt.fmt.pix.sizeimage);
// draw line
p = (char*)capt.buf.m.userptr;
for(int i=0; i<21600; i++){
*(p+i) = 0xFF;
}
/* Exchange display and capture buffer pointers */
temp_buf.m.userptr = capt.buf.m.userptr;
capt.buf.m.userptr = disp.buf.m.userptr;
disp.buf.m.userptr = temp_buf.m.userptr;
/* Queue the capture buffer with updated address */
ret = ioctl(capt.fd, VIDIOC_QBUF, &capt.buf);
if (ret < 0) {
return (-1);
}
disp.buf.type = V4L2_BUF_TYPE_VIDEO_OUTPUT;
disp.buf.memory = V4L2_MEMORY_USERPTR;
disp.buf.length = capt.fmt.fmt.pix.sizeimage;
/* Queue the display buffer back with updated address */
ret = ioctl(disp.fd, VIDIOC_QBUF, &disp.buf);
if (ret < 0) {
return (-2);
}
Source code base
saLoopBack.c
varinfo.xres = disp.fmt.fmt.pix.width;
varinfo.yres = disp.fmt.fmt.pix.height;
varinfo.xres_virtual = varinfo.xres;
varinfo.yres_virtual = varinfo.yres * VIDEO_MAX_BUFFER;
// VIDEO_MAX_BUFFER=8
varinfo.bits_per_pixel = 24;
varinfo.transp.offset = 0;
varinfo.transp.length = 0;
varinfo.transp.msb_right= 0;
ret = ioctl(fbdev_fd, FBIOPUT_VSCREENINFO, &varinfo);
if (ret < 0) {
return (-4);
}
buffersize = fixinfo.line_length * varinfo.yres;
buffer_addr[0] = (unsigned char *)mmap(0,
buffersize * VIDEO_MAX_BUFFER,
(PROT_READ | PROT_WRITE),
MAP_SHARED, fbdev_fd, 0);
for (i = 1; i < VIDEO_MAX_BUFFER; i++)
buffer_addr[i] = buffer_addr[i-1] + buffersize;
memset(buffer_addr[0], 0x00, buffersize * VIDEO_MAX_BUFFER);
/* Set format according to display mode */
disp.fmt.fmt.pix.width = VIDEO_IMAGE_WIDTH;
disp.fmt.fmt.pix.height = VIDEO_IMAGE_HEIGHT;
disp.fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;
disp.fmt.fmt.pix.bytesperline = disp.fmt.fmt.pix.width * 2;
disp.fmt.fmt.pix.sizeimage = disp.fmt.fmt.pix.bytesperline *
disp.fmt.fmt.pix.height;
ret = ioctl(disp.fd, VIDIOC_S_FMT, &disp.fmt);
if (ret < 0) {
qDebug("Set Format failed");
return (-2);
}