Vybrid Family VF6XX Linux Framebuffer and Display System Architecture and User Guide

Legend

🟨 - Highlighted in yellow, needs Freescale input
⬜ - Highlighted in grey, TBD/InProgress/Release 2

Requirements

The requirements document provides the features supported by the linux display system driver for the Freescale MVF platform.

The MVF linux BSP requirements document is at the link below:
https://www.freescale.com/livelink/livelink/225971157/Linux_BSP_SRS_Faraday_v0.02-20120330.xlsx?func=doc.Fetch&nodeid=225971157

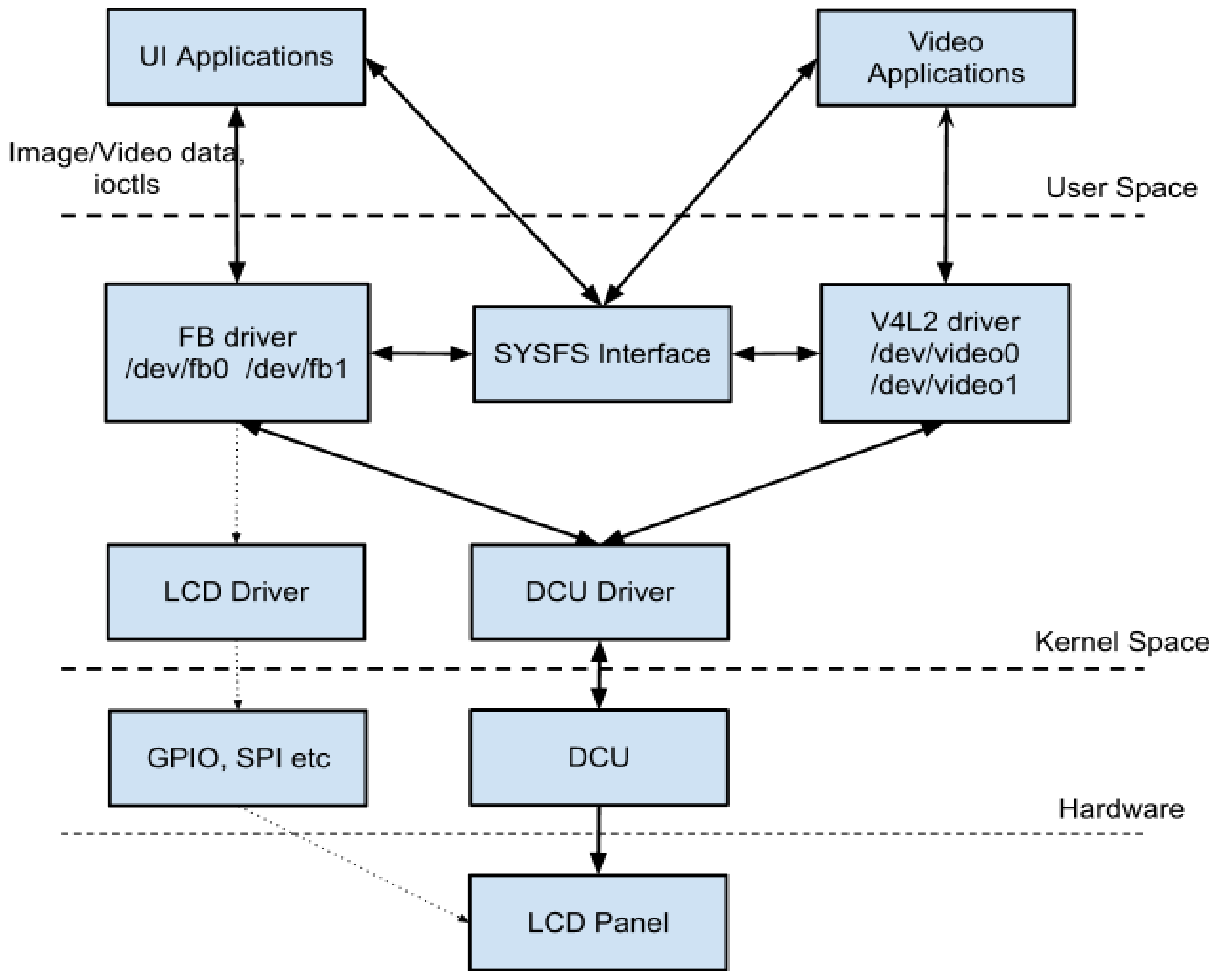
The display system only requirements extracted from the above document along with planned release timeline is located at:
https://docs.google.com/spreadsheet/ccc?key=0AjnVFUZY1JDddDV5UDFuNDdaSHZQVEVoSFNqdGVzcXc

Introduction

This document provides details on configuring and using the display driver on the Linux platform for Freescale Vybrid family VF6xx. It also describes the driver structure and design.

Architecture

The VF600 chip supports upto 2 simultaneous displays handled by the Display Controller Units DCU0 and DCU1. The display interface is 24 bit RGB interface. The linux BSP for MVF platform contains four drivers for configuring the display system. The Framebuffer (FBDev) driver, LCD driver, VideoForLinux (V4L2) driver and the DCU driver.



Driver configuration and usage

The FBDev, LCD and V4L2 drivers can be configured as static or loadable modules. The DCU driver is static driver only. When LCD driver is configured as loadable module, either FBDev or V4L2 driver needs to be loaded before loading the LCD Driver.

Kernel build configuration

To configure the drivers, user needs to select the Configure Kernel option in the main menu when executing:
    ./ltib --config
    <*> Configure Kernel
Once the user saves configuration and exits the ltib menu, the Kernel configuration menu pops up and the user needs to select the appropriate drivers as shown below to get display system working in Linux.

In the main menu of Kernel Config select the Device Drivers option:

- Device Drivers > Graphics support
    - Support for frame buffer devices [FB]
        - Number of Framebuffers for Overlay (1) [FB_MVF_NUM_FB_OVL]
        - Enable video mode handling helpers (needed by EDID?) [FB_MODE_HELPERS]
    - Backlight and LCD support [BACKLIGHT_LCD_SUPPORT]
        - Low level backlight controls [BACKLIGHT_CLASS_DEVICE]
            - Generic PWM based backlight driver [BACKLIGHT_PWM]
    - MVF Framebuffer support [FB_MVF]
        - MVF EDID support [FB_MVF_EDID]
        - Panel Framebuffer [FB_MVF_PANEL]
            - NEC WQVGA Panel [FB_MVF_NEC_WQVGA_PANEL]
    - Bootup Logo support (w/ all sub options)

For DCU driver configuration:

- Device Drivers > MVF support drivers
    - Display Controller Unit Driver [MVF_DCU_V4]

    For V4L2 driver configuration:

- Device Drivers > Multimedia support
    - Video for Linux [VIDEO_DEV]
    - Video capture adapters [VIDEO_CAPTURE_DRIVERS]
        - MVF Video For Linux Camera [VIDEO_MVF_CAMERA]
            - Omnivision camera support [MVF_CAMERA_OV3640]
        - MVF Video for Linux Output [VIDEO_MVF_OUTPUT]
            - MVF V4L2 support [VIDEO_MVF_DCU_OUTPUT]

Note: Each overlay will be enumerated as a unique framebuffer device. Number of Framebuffer for Overlays is a range of 0 - 30. The number of overlays is for both DCU's together.

Kernel command line options via bootargs

The display bootargs options for the FB driver built-in to the kernel are as follows:
video=<Desired_DCU>:<Pixel_Format>,<Desired_LCD> <Primary_Display> fbmem=<size>[@<physaddr>]
where
<Desired_DCU> options are:

- mvfdcu0fb
- mvfdcu1fb

    <Pixel_Format> options are:

- RGB565

- RGB888
- BGRA8888
- YCbCr422

    <Desired_LCD> options are:

- NEC-WQVGA

    <Primary_Display> options are:

- dcu0_primary
- dcu1_primary

    fbmem: Pool of memory in external RAM to preallocate for Framebuffers and optionally a physical start memory address. Memory reserved for all framebuffers in the system inclusive of overlays and double buffering. “fbmem=” boot argument is optional when using single buffer scheme. See memory allocation FAQ’s for more details.

- <size>
    - Size of Framebuffer memory to be reserved (M in megabytes or K in kilobytes)
- <physaddr>
    - If physaddr is zero, memory will be allocated at the start of external RAM address (if available)
    - If physaddr is non-zero, the memory will get allocated starting at Physical Address “physaddr” (if available).

    Example usage: fbmem=10M@0x80000000

    For the TWR-VF600 board with NEC display connected to DCU0 use the below bootargs:
    video=mvfdcu0fb:BGRA8888,NEC-WQVGA dcu0_primary

    Note: When the “video=” boot argument is not provided, the frame buffer devices will not be available. If “Primary_Display” argument is not provided, the first DCU defined in the boot argument will be defaulted to /dev/fb0.

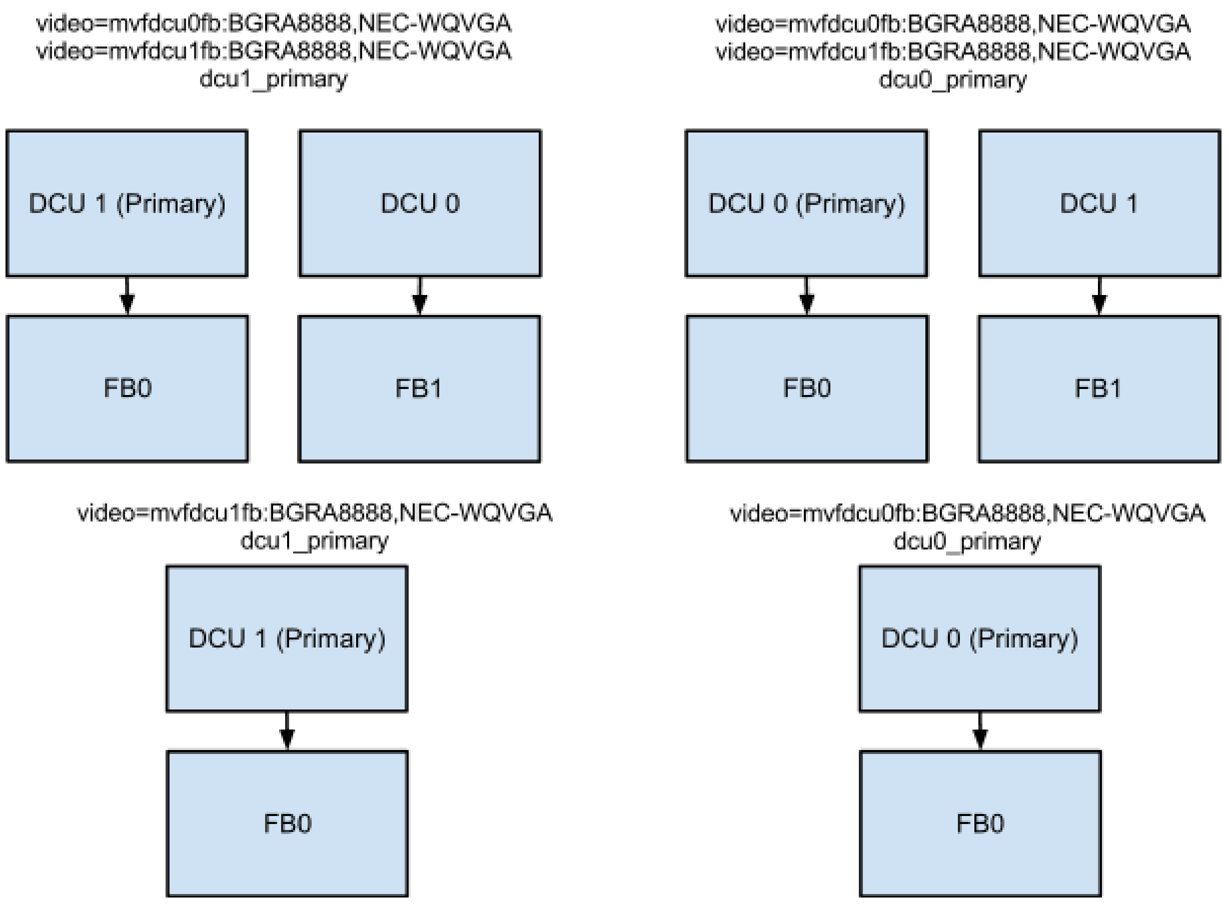
    Supported formats for first release:
    V4L2_PIX_FMT_RGB24
    V4L2_PIX_FMT_RGB32
    V4L2_PIX_FMT_RGB565

Install module options

insmod mvf_fb.ko

Framebuffer Configurations:

DCU to Frame Buffer Mapping Use Cases:


video=mvfdcu0fb:BGRA8888,NEC-WQVGA
video=mvfdcu1fb:BGRA8888,NEC-WQVGA
dcu1_primary
DCU 1 (Primary)
DCU 0
FB0
FB1
video=mvfdcu0fb:BGRA8888,NEC-WQVGA
video=mvfdcu1fb:BGRA8888,NEC-WQVGA
dcu0_primary
DCU 0 (Primary)
DCU 1
FB0
FB1
video=mvfdcu1fb:BGRA8888,NEC-WQVGA
dcu1_primary
DCU 1 (Primary)
FB0
video=mvfdcu0fb:BGRA8888,NEC-WQVGA
dcu0_primary
DCU 0 (Primary)
FB0

Memory Allocation:

Double buffering of the frames is supported which allows displaying data from one buffer while the other buffer is filled with new data. The size of the frame buffer depends on the display size, number of desired layers, double buffering option and must be calculated. If “fbmem=” boot argument is provided then double buffering is automatically enabled for the framebuffers defined at boot. The user has an option to enable Double buffering for each overlay.

Device	bootarg “video=” not provided	bootarg “fbmem=” not provided	Both, fbmem= and video= provided
FB0(dev/fb0)	0 bytes	(Display Res) * BytesPerPixel	(Display Res) * BytesPerPixel * 2
FB1(dev/fb1)	0 bytes	(Display Res) * BytesPerPixel	(Display Res) * BytesPerPixel * 2
Overlay FB2 - FB32 (dev/fb2 .. dev/fb32)	N/A	N/A	0 bytes by default. Memory is only allocated when the respective overlay framebuffer device is opened in user

			application.

Framebuffer driver driver allocates one physically contiguous buffers for each node, which can support up to 800x480 resolution 32 bpp format.

In general the memory allocated for a framebuffer device is:
*Frame buffer size = XResolution * YResolution * BytesPerPixel * NumberOfBuffers*
where
XResolution - Horizontal resolution of the layer
YResolution - Vertical resolution of the layer
BytePerPixel - Guess what? :-)
NumberOfBuffers - 1 for Single Buffering, 2 for Double Buffering.

Memory allocation use case analysis:

Single layer
Settings: Double buffer desired, Size of layer = 480 * 272, Pixel Format = BGRA8888
Frame Buffer Size = 480 * 272 * 4 * 2 = 1020kB

Two layers
Settings: Double buffer desired, Size of Layer1 = 480 * 272, Size of Layer2 = 320 * 240 Pixel Format = BGRA8888
Frame Buffer Size = (480 * 272 * 4 + 320 * 240 * 4) * 2 = 1620kB

Generic notes:
- The entire memory allocation for framebuffer is done on external RAM (external to the VFxxx chip)

User application API's

Once the driver is loaded the hardware can be accessed using special nodes (like any other character device) that are located in the /dev directory of the root.

FB driver usage

File operations
The following file operations are available to the user:

- open - Open the framebuffer device
```
int fbfd;
// Open the file for reading and writing
fbfd = open("/dev/fb0", O_RDWR);
if (fbfd == -1) {
        perror("Error: cannot open framebuffer device");
        return -1;
}
```

- map - Makes the memory allocated for the framebuffer accessible to user space. The data written to the framebuffer memory will be displayed on the LCD panel.
```
        char *fbp;
// Map the device to memory
// SCREEN_SIZE = finfo.line_length * vinfo.yres; -> Obtained via ioctl operations illustrated later
fbp = (char *)mmap(0, SCREEN_SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, fbfd, 0);
if ((int)fbp == -1) {
      perror("Error: failed to map framebuffer device to memory");
      return -1;
        }
```

- close - Close the frame buffer device
```
        close(fbfd)
```

- fb_setcolreg
- fb_setcmap - Set CLUT memory
- fb_cursor - Set cursor image

NOTES:
- fb_fillrect, fb_copyarea, fb_imageblit (mainly used by console/display logo routines) will be remapped to standard software functions cfb_fillrect, cfb_copyarea, cfb_imageblit and will not be HW accelerated
- fb_rotate, fb_read, fb_write, get_unmapped_area will not implemented

ioctl operations
The following ioctl operations are available to the user:

- FBIOGET_FSCREENINFO - Gets the fixed screen information about the framebuffer device.
```
struct fb_fix_screeninfo finfo;
// Get fixed screen information
if (ioctl(fbfd, FBIOGET_FSCREENINFO, &finfo) == -1) {
        perror("Error reading fixed information\n");
        return -1;
}
```

- FBIOGET_VSCREENINFO - gets the variable screen information about the framebuffer device.
```
struct fb_var_screeninfo vinfo;
// Get variable screen information
if (ioctl(fbfd, FBIOGET_VSCREENINFO, &vinfo) == -1) {
      perror("Error reading variable information\n");
```

```
        return -1;
    }
```

- FBIOPUT_VSCREENINFO - sets the variable screen information.

```
    // Modify information obtained from FBIOGET_VSCREENINFO and write it back
    if (ioctl(fbfd, FBIOPUT_SCREENINFO, &vinfo) == -1) {
        perror("Error writing variable information\n");
        return -1;
    }
```

- FBIOBLANK - blank or unblank display. <ADD what exactly happens to the vsync/hsync/pixclk signals on the pins>

```
            //User options:  FB_BLANK_UNBLANK or FB_BLANK_POWERDOWN
    if (ioctl(fbfd, FBIOBLANK, FB_BLANK_POWERDOWN) == -1) {
        perror("Error performing blank operation\n");
        return -1;
    }
```

Recommended implementation from Freescale:

1. Turn off backlight
2. Write DCU_MODE_DCU_MODE to 0 to stop updating
3. Write DCU_MODE_RASTER_EN to 0 to stop HSYNC& VSYNC
4. Set IOMUX107 to GPIO to remove DCU clock

- FBIOPAN_DISPLAY - Double buffering can be achieved by the same.

```
    //Get VarScreen Info, modify yoffset to point to buffer 2 and update buffers.
    ioctl(fbfd, FBIOGET_VSCREENINFO, &fbvar);
    fbvar.yoffset = 480;
    ioctl(fbfd, FBIOPAN_DISPLAY, &fbvar);
    //to go back to buffer 1.
    fbvar.yoffset = 0;
    ioctl(fbfd, FBIOPAN_DISPLAY, &fbvar);
```

- MVFFB_ENABLE_OVERLAY_DOUBLE_BUFFER - Enables / disables double buffering for desired overlays.

```
    //Enable / disable double buffering for each layer individually. Enable - TRUE, Disable - FALSE
    bool flag = TRUE;
    if (ioctl(fbfd, MVFFB_ENABLE_DOUBLE_BUFFER, flag) == -1) {
        perror("Error allocating memory\n");
        return -1;
    }
```

- MVFFB_MAP_OVERLAY_TO_DISP - Maps the respective overlay to the desired display.

```
    //Map each layer individually to the desired display.
    enum{
    PRIMARY
    SECONDARY
    }
    if (ioctl(fbfd, MVFFB_MAP_OVERLAY_TO_DISP, PRIMARY) == -1) {
        perror("Error mapping to display\n");
        return -1;
    }
```

- MVFFB_WAIT_FOR_VSYNC - Waits for next Vertical Sync.

```
    int arg = 0;
    if (ioctl(fbfd, MVFFB_WAIT_FOR_VSYNC, &arg) == -1) {
        perror("Error getting Vertical Sync\n");
        return -1;
    }
```

- FBIOPUTCMAP / FBIOGETCMAP
- FBIOPUT_CON2FBMAP / FBIOGET_CON2FBMAP
- MVFFB_SET_GBL_ALPHA
- MVFFB_SET_CLR_KEY
- MVFFB_SET_LOC_ALPHA
- MVFFB_SET_GAMMA

Sysfs Interface [Only for Release 2]

- Blank / UnBlank display - echo 0 > /sys/class/graphics/fb0/blank [0,1]
- Rotate - echo 0 > /sys/class/graphics/fb0/rotate [0,1,2,3]
- Reading FB Attributes - cat /sys/class/graphics/fb0/phys_addr [phys_addr, virt_addr, size]
- Enable / Disable display - echo 1 > /sys/devices/platform/lcd/enabled
- Change Timing for LCD - <pixel_clock>,<xres/hfp/hbp/hsw>,<yres/vfp/vbp/vsw>.
    - eg: echo "13500,720/16/58/64,482/6/31/6" > /sys/devices/platform/lcd/timings

Driver Design

This section is intended for understanding the MVF display subsystem driver design.

Linux kernel paths:

The path for the display subsytem drivers is as follows:

- DCU driver location:
    - /drivers/mvf/dcu4
        - dcu4_regs.h

        - dcu4_driver.c
        - dcu4_driver.h
- MVF Framebuffer and LCD panel driver location:
    - /drivers/video/mvf
        - mvf_dcu4_fb.c
        - mvf_nec_wqvga.c
        - mvf_fb_modedb.c
- Backlight control
    - /drivers/video/backlight
- General MSL for Vybrid family
    - /arch/arm/mach-mvf/board-twr_vf600.c
    - /arch/arm/plat-mxc/include/mach/dcu4.h

LCD panel support

TWR-VF600 LCD interface

The TWR-VF600 board interfaces with TWR-LCD-RGB board which inturn has a connector to interface with a TFT LCD. Details on the hardware signals between the Vybrid chip and LCD connector is as follows:
- LCD_D/P[0-23] -> Display RGB[0-8]
- LCD_HSYNC/P24 , LCD_VSYNC/P25, LCD_CLK/P26 -> Display HSYNC/VSYNC/CLK
- LCD_OE/P27 -> Display RGB_DE
- RESET_B (System reset and SW1)
- PWM15 (LED_PWM) OR GPIO22 (LCD_ENABLE/PTB8/) via J9 -> Backlight controller -> Display LCD_LED1_A/ LCD_LED2_A
    - Note: PTB8 is being used as FTM1CH0 (PAD30) which is a FlexTimer used as PWM
    - LED_PWM is not connected on the Vybrid board
- ADC AN8-11 <- Resistive touch input via jumper OR (I2C via IRQ_I or IRQ_K) <- Touch controller CRTOUCHB10VFM

NEC **NL4827HC19-05B display**

The TWR-LCD-RGB board is designed to work with a NEC display (NL4827HC19-05B). The critical specs from the LCD datasheet are as follows:
- 480x272 RGB with touch capability
- Typical refresh rate is 75 Hz
- Pixel clock - Min: 8.69 Typ: 10.87 Max: 11.59 (MHz)
- HYSNC - Front/Back porch : 2 pixel clocks, Pulse width: 41 pixel clocks (min: 2)
- VSYNC - Front/Back porch : 1H, Pulse width: 2H (min: 1) where 1H = HPulse+ FP+ W + FH = 525 pixel clocks
- HSYNC/VSYNC: Active low

NEC panel support in linux

Based on the above specs the modedb structure in Linux which contains the information about the display panel can be initialized as below:

```
struct fb_videomode {
    const char *name;      /* optional */
    u32 refresh;           /* optional */
    u32 xres;
    u32 yres;
    u32 pixclock;
    u32 left_margin;
    u32 right_margin;
    u32 upper_margin;
    u32 lower_margin;
    u32 hsync_len;
    u32 vsync_len;
    u32 sync;
    u32 vmode;
    u32 flag;
};

struct   mvffb_modedb[] = {
    {
     /* 480x272 @ 75 Hz */
     "NEC-WQVGA", 75, 480, 272, KHZ2PICOS(10870), 2, 2, 1, 1, 41, 2,
    /*Active low HSYC/VSYNC, Sample on falling edge of pxl clk, output not negated*/
     0,
     FB_VMODE_NONINTERLACED,
     0,},
}
```

Custom panel support

The LCD driver reserves all 24 data pins for the display interface on TWR-VF600 board. If the display does not need all 24 lines then these pins can be freed for other functionality by editing the file /arch/arm/mach-mvf/

If a custom LCD panel support needs to be added then ....

Framebuffer resources

- linux/documentation/fb
- linux/include/linux/fb.h

- linux/drivers/video/skeletonfb.c
- linux/drivers/video/fbmem.c
- linux/drivers/video/vfb.c
- linux/drivers/video/mxc/ mxc_ipuv3_fb.c and mxcfb.c
- linux/drivers/video/fsl_diu_fb.c
- http://cache.freescale.com/files/dsp/doc/app_note/AN4182.pdf
- http://www.atmel.com/Images/doc32105.pdf
- http://processors.wiki.ti.com/index.php/DM814X_AM387X_VPSS_Video_Driver_User_Guide

Hardware/Software used for testing

- TWR-VF600 Rev:
- TWR_ELEV Rev:
- TWR-SER Rev:
- TWR-LCD-RGB with NL4827HC19-05B LCD Rev:
- Freescale J-Link Lite Rev:

Blocking items / Dependencies:

- For Release 2
  - Segmented LCD interface and display + datasheet
  - Board which supports 2 TFT LCD's
  - Video source and interface (VIU and PDI)??
  - CAM_00010 - The V4l2 input device driver shall support the OV3640 camera module
    - Timesys supposed to work? VIU not part of SOW? Need hardware.
  - LCD display without built-in timing control to test TCON

TWR-VF600 limitations and testing implications

- Segmented LCD, Video ADC, VIU, 2nd DCU, DCU TCON signals are not supported on TWR-VF600 board **- So can not be tested**

Closed items

- FB_00032 - The framebuffer driver shall support the eLCD interface.
  - We only have DCU, there is no eLCD interface like in i.MX502/503. Requirement invalid.
    - Freescale Response: FB_00032 - does not apply to you.
- FB_00255 - The V4L2 output device shall support scaling
  - DCU does not support scaling. Scaling will be in software and not hardware accelerated
    - Freescale Response: That is expected.
- FB_00210 - The V4L2 output device driver shall support rotation (0, 180, +/-90)
  - DCU does not support rotation. Rotation will be in software and not hardware accelerated
    - Freescale Response: That is expected.
- FB_00105 - The framebuffer driver shall support the following pixel formats for all displays: RGB565, BGR888, RGB666
  - RGB666 format is not supported by DCU. BGR888 format is not natively supported by DCU, requirement could be RGB888 or we might have to do a swap at the TCON layer
    - Freescale Response: Please support what you can for Rev 1. I thought I pulled these from the Reference manual but it was an earlier version.
- FB_00300 - The frame buffer driver shall provide an ioctl to return the list of supported resolutions and frequencies to user space. If the display supports multiple resolutions, EDID functions shall be used. Otherwise, the static resolution and frequencies shall be returned.
  - Are we expected to get the EDID information over I2C? Are there any displays which support EDID and interface with TWR-VF600?
    - Freescale Response: FB_00300 - An LCD display does not support EDID. The "Otherwise" clause is the one of interest.
- The below features are not part of the requirements, do they need to be supported?
  - LCD backlight brightness control
    - The backlight is controlled by PWM. Which is in the SOW/requirements (it's just one line) for Rel 1. It should be controlled through framebuffer.
  - Display support in uBoot
    - No. We do not need display support in u-boot
  - PDI module of DCU
    - Freescale Response: This is not needed for Rel 1. In a later release it could be added.
- Touch screen driver is not part of SOW, is Timesys supposed to work on this?
  - Freescale Response: For the touchscreen, I believe that was going to be done by someone here.
- FB_00710 - FB_00717: Segmented display support
  - TWR-VF600 hardware does not support segmented LCD
    - Freescale response: As for the Segmented LCD, I think that ended up being the TFT display.