LSP 1.20 DaVinci Linux Frame Buffer Driver Migration

ABSTRACT

This migration document provides information to developers who will be working with the LSP 1.20 Frame Buffer (FB) device for the TI DaVinci Video Processing Back End (VPBE) subsystem. The FB device driver has been redesigned since version LSP 1.10 to meet new system requirements.

Contents

List of Tables

1 Terms and Abbreviations

Table 1. Terms and Abbreviations

Term	Description
API	Application Programming Interface
FB	Frame Buffer
OSD	On Screen Display
IOCTL	Input Output Control
VPBE	Video Processing Back End
LSP	Linux Support Package

2 Background

In LSP1.10, the video driver for TI DaVinci products consists only of the Linux frame buffer device driver. This driver's API is designed around the scenario that the frame buffer dimensions and the display screen dimensions are one and the same. For example, the xres and yres values in the fb_var_screeninfo not only define the size of the image in memory, they also define the resolution of the display. The display

timing values (pixclock, left_margin, right_margin, upper_margin, lower_margin, hsync_len, and vsync_len) are also specified in the fb_var_screeninfo. The underlying assumption is that these values can always be used together with xres and yres to calculate all of the display timing settings that would be needed to initialize the display controller. In addition, each FB device instance (/dev/fb/0, /dev/fb/1, etc) is implicitly assumed by the API design to be completely independent of all other instances.

The DaVinci video controller is designed to support four video windows (OSD0, OSD1, VID0, and VID1), which are implemented in the FB driver as independent FB device instances. However, there are actually four windows of the same display, not four independent displays. The xres and yres values in the fb_var_screeninfo are interpreted as the dimensions of the corresponding window, which are not necessarily the same as the dimensions of the display. In fact, there is no way to specify the display dimensions via the standard FB API.

The fact that the frame buffer dimensions are not the display dimensions leads to a problem with specifying the display timing. You must know the display resolution in order to calculate the display timing, but since the display resolution is unknown within the confines of the standard FB API, it is impossible to set the display timing via the usual method.

Within DaVinci, the fact that all four windows share a common underlying display leads to conflicts with the FB API. For example, on the DaVinci processor, the display timing is shared by all windows, but the FB API will allow an application to set unique timing values on a per-device basis.

Additionally, the DaVinci video windows use a YUV pixel format rather than an RGB format. The Linux FB API only supports RGB format. Therefore, the FB devices for the video windows will only work with applications specifically written for DaVinci. When the OSD1 window is used as an attribute window, only applications written specifically for DaVinci will be able to utilize it.

Given the above limitations of the Linux FB API as applied to the DaVinci display controller, it was unavoidable that some non-standard mechanisms must be implemented to allow applications to fully utilize the DaVinci display controller.

The LSP 1.10 FB driver implementation defines 27 non-standard IOCTLs. The LSP1.20 has a few improvements in the driver, like removing some redundant IOCTLs and providing alternatives of using these non-standard IOCTLs.

3 LSP1.20 IOCTL Changes

In LSP1.20, Linux sysfs attributes supporting the FB device driver have been implemented as an alternative to or replacement for the IOCTLs. Sysfs is a virtual file system that is normally mounted at /sys in a Linux root file system. A device driver can create files within sysfs that can be used to display (by reading the file) or modify (by writing to the file) arbitrary parameters associated with the driver. For the DaVinci VPBE, the sysfs attributes are located in: /sys/class/davinci_display directory.

There are two important distinctions between sysfs driver attribute files and IOCTLs. First, IOCTLs are associated with a specific FB device (OSD0, OSD1, VID0, and VID1 for DaVinci), but driver attribute files are global and are not associated with a particular frame buffer device. Second,the IOCTLs require custom applications to access them, and there is no method to enumerate the controls that are available. On the other hand, driver attribute files can be accessed with standard shell commands (e.g. 'cat < /sys/attrfile' to read and 'echo xyz > /sys/attrfile' to write). The available attributes can be enumerated simply by listing the sysfs directory where the attributes are defined.

As a result of the new design, a few of the non-standard IOCTLs can be deprecated; their functions can be replaced by either using the FB driver API or by accessing sysfs attributes.

Table 2 lists those deprecated IOCTLs that are retained for backward compatibility. They can be replaced by using the alternative API or sysfs attributes mentioned in the Notes column. For more detailed information on using the alternatives, please refer to the LSP1.20 Frame Buffer Device Driver User's Guide.

Table 2. Deprecated IOCTLs

Deprecated IOCTL Name	Use	Notes
FBIO_SETATTRIBUTE	Set the blend factor for a rectangular area in the attribute window	The application can write attribute pixel values directly to the rectangular area in the OSD1 frame buffer.
FBIO_ENABLE_DISABLE_WIN	Enable or disable the display of a window	Use FBIOBLANK IOCTL instead. An additional way is implemented in the FB driver to enable and disable the window. If FBIOPUT_VSCREENINFO ioctl is used with any xres, yres, xres_virtual, yres_virtual set to zero, then the window will be disabled. It can be reenabled by using FBIOPUT_VSCREENINFO again and setting a valid video mode.
FBIO_SET_BITMAP_WIN_ RAM_CLUT	Load values into the RAM CLUT (color lookup table).	Use FBIOPUTCMAP IOCTL with RGB format. FB driver converts it to YUV format internally.
FBIO_ENABLE_DISABLE_ ATTRIBUTE_WIN	Enable or disable the attribute window functionality of OSD1	Attribute mode can be enabled via the standard FBIOPUT_VSCREENINFO by setting bits_per_pixel to 4 and nonstd to non-zero. It can be disabled by setting nonstd to zero. When enabling attribute mode, var->bits_per_pixel is set to 4. var->xres, var->yres, var->xres_virtual, var->yres_virtual, win->xpos, and win->ypos are all copied from OSD0. var->xoffset and var->yoffset are set to 0. fix->line_length is updated to be consistent with 4 bits per pixel. No changes are made to the OSD1 configuration if OSD1 is already in attribute mode. When disabling attribute mode, the window geometry is unchanged. var->bits_per_pixel remains set to 4. No changes are made to the OSD1 configuration if OSD1 is not in attribute mode.

Table 3 lists those deprecated IOCTLs that are no longer supported.

Table 3. Removed IOCTLs

Deprecated IOCTL Name	Use	Notes
FBIO_SET_INTERFACE	Set the output display interface	Display output switch is replaced by writing into sysfs attribute /sys/class/davinci_display/ch#/output where ch# is the channel number of the display on the system.
FBIO_GET_INTERFACE	Get the output display interface	Getting display output is done by reading sysfs attribute /sys/class/davinci_display/ch#/output
FBIO_GETSTD	Get the current video standard	Getting display standard is done via reading sysfs attributes /sys/class/davinci_display/ch#/mode
FBIO_QUERY_TIMING	Querying video mode definition	Mode info can be retrieved via FBIOGET_VSCREENINFO For further details, please see LSP1.20 FB Device Driver User's Guide
FBIO_SET_TIMING	Given a video mode definition, switch to a display that supports the mode	Can use FBIO_PUT_VSCREENINFO IOCTL to set non-standard timings. For further details, please see LSP1.20 FB Device Driver User's Guide
FBIO_GET_TIMING	Get the current video mode definition	Can be replaced with new IOCTL: FBIOGET_VSCREENINFO

Table 3. Removed IOCTLs (continued)

Deprecated IOCTL Name	Use	Notes
		For further details, please see LSP1.20 FB Device Driver User's Guide
FBIO_SET_VENC_CLK_SOURCE	Get the VENC clock source	Use sysfs attribute instead (this attribute is not yet implemented)
FBIO_ENABLE_DISPLAY	Enable/disable the display	Use sysfs attribute instead /sys/class/davinci_display/ch0/enable

Table 4 shows the slightly modified behavior of some of the retained IOCTLs in LSP1.20.

Table 4. Changed IOCTLs

Deprecated IOCTL Name	Use	Notes
FBIO_GET_VIDEO_CONFIG_PARAMS FBIO_SET_VIDEO_CONFIG_PARAMS	Get/set video parameters	When changing the Cb/Cr order, the change will not take effect until an FBIOPUT_VSCREENINFO ioctl is issued for a window with a YUV pixel format. And this change is global, meaning all other display windows in YUV format get switched at the same time.
FBIIO_SETZOOM	Setting zooming of the display	The 'window_id' parameter is now ignored since this ioctl is issued to a specific FB device.

4 LSP1.20 Kernel Boot Arguments for the Frame Buffer Driver

LSP1.20 Kernel Boot Argument format for the Frame Buffer Driver is as follows:

- *video = [davincifb | dm64xxfb | dm355fb]* – dm64xxfb and dm355fb are deprecated, *davincifb* should be used instead.
- *vid0=[off | MxNxP, S@X,Y]*
- *vid1=[off | MxNxP, S@X,Y]*
- *osd0=[MxNxP, S@X,Y]*
- *osd1=[MxNxP, S@X,Y]*

where MxN are the horizontal and vertical window size; P is the color depth (bits per pixel), S is the frame buffer size in bytes with suffixes such as 'K' or 'M' for Kilo (2^10) and Mega (2^20); X, Y are the window position. Only video windows can be turned off. Turning off a video window means that no FB device will be registered for it.

For example:

```
video=davincifb:vid0=720x480x16,2025K@0,0:vid1=720x480,1350K@0,0:osd0=720x480,1350K@0,0:osd1=720x480,1350K@0,0
```

In the above example, the vid0 is reserved with buffer size 2025K bytes, which is large enough for triple buffering at 720x280x16. The osd0 window is reserved with buffer size 1350K, which is for double buffering for 720x480x16. The FB driver limits video windows to triple buffering and osd windows to double buffering. The total size of the buffer for all display windows shall not exceed 40M bytes.

Specific window can be disabled using boot argument option as below:

```
video=davincifb:vid0=off:vid1=off

or

video=davincifb:vid0=0,0:vid1=0,0
```

In this example both the vid0 and vid1 will be disabled at boot time. This will prevent the FBDev driver

from creating devices for vid0 and vid1 (/dev/fb/1 and /dev/fb/3). If any of the windows is disabled at boot time, any FBDev driver application is not allowed to perform any IO control operation with that window. However, this will allow other video applications (e.g. V4L2) to access the video windows disabled by FB driver bootargs. OSD windows, however, cannot be disabled by boot arguments. Even if setting up an OSD window as “off” in the bootargs, it will be ignored by FB driver and set it up with default values.

When a video window is turned off at boot time, no /dev/fb or /proc/dev entry will be created for it.

Alternatively, the following boot arguments can be used to prevent the FBDev driver from claiming video windows, but still reserve the frame buffer space and create FBDev devices. In other words, this will allow V4L2 applications to access vid0 and vid1 windows, yet FBDev devices /dev/fb/1 and /dev/fb/3 will still be created.

```
video=davincifb:vid0=0,2025K:vid1=0,1350K
```

After booting up, all FB devices are created as normal, and V4L2 applications are able to claim video windows (through /dev/video/2 or /dev/video/3) to use. When an FBDev application needs to use the device, use ‘fbset’ command to allow FBDev driver to re-claim the video windows (to desired resolution):

```
$ fbset -fb /dev/fb/1 -xres 720 -yres 480 -vxres 720 -vyres 1440 -depth 16
$ fbset -fb /dev/fb/3 -xres 720 -yres 480 -vxres 720 -vyres 1440 -depth 16
```

A second alternative is, instead of disabling these windows using boot arguments, using ‘fbset’ to release the windows from FBDev driver’s control for other applications to use even if FBDev devices are enabled at boot argument. The following example shows two console commands to “turn off” osd0 and vid0 windows, respectively.

```
$ fbset -fb /deb/fb/0 -xres 0
$ fbset -fb /dev/fb/1 -xres 0
```

When these display windows need to be used by an FBDev application, use ‘fbset’ again to restore the frame buffer device. The following example shows the command of setting vid0 display window to NTSC window size with triple buffering.

```
$ fbset -fb /dev/fb/1 -xres 720 -yres 480 -vxres 720 -vyres 1440
```

5 Setting Up Frame Buffer Size

To set the bootarg to the LSP1.10 FB default behavior, use the following settings:

```
video=davincifb:osd0=720x480x16,1350K:osd1=720x480,1350K:vid0=720x480,2025K:vid1=720x480,2025K
```

These settings will make the OSD frame buffers large enough for double buffering at 720x480x16, and the video frame buffers large enough for triple buffering at 720x480x16. Now, however, the FB driver always makes the initial window resolution (xres, yres) and frame buffer resolution (xres_virtual, yres_virtual) match, so the extra buffers in the frame buffer won't be accessible until an fbset command (or equivalently an FBIOPUT_VSCREENINFO ioctl) is used to increase yres_virtual to allow for screen flipping.

The virtual resolution (var->yres_virtual) in the application has to be set explicitly to support double or triple buffering, provided that the frame buffer memory allocation is large enough. Otherwise FBIOPUT_VSCREENINFO ioctl will fail.

In order to make legacy applications work as-is (without fixing them), the following fbset commands can be used before running the application. These commands assume that an NTSC display is used and the frame buffers are sufficiently large to accommodate these sizes.

Make OSD0 720x480x16 with double buffering:

```
fbset -fb /dev/fb/0 -xres 720 -yres 480 -vxres 720 -vyres 960 -depth 16 -nonstd 0
```

Make VID0 720x480x16 (YCbCr) with triple buffering:

```
fbset -fb /dev/fb/1 -xres 720 -yres 480 -vxres 720 -vyres 1440 -depth 16 -nonstd 1
```

Make OSD1 720x480x4 (attribute mode) with double buffering:

```
fbset -fb /dev/fb/2 -xres 720 -yres 480 -vxres 720 -vyres 960 -depth 4 -nonstd 1
```

Make VID1 720x480x16 (YCbCr) with triple buffering:

```
fbset -fb /dev/fb/3 -xres 720 -yres 480 -vxres 720 -vyres 1440 -depth 16 -nonstd 1
```

Also note that the virtual horizontal size can be made larger than the window's horizontal size in order to accommodate displaying an image that was created with a longer line length than the display. For example, if a static image was created for a 720x480 display, it can be displayed in a 640x480 window by doing the following:

```
fbdev -fb /dev/fb/3 -xres 640 -vxres 720 -yres 480 -vyres 480
cp my_720x480_image /dev/fb/3
```

In this case, the 80 pixels at the end of every line will not be displayed.

6 Setting Output and Mode

In LSP1.20, output interface parameters set by user are passed to Encoder Manager for processing. They can be set through Encoder Manager boot argument as follows at boot time:

```
davinci_enc_mngr.ch0_output=COMPOSITE davinci_enc_mngr.ch0_mode=NTSC
```

Or, after the kernel boots up, they can be set by writing the output string and mode string into two Davinci sysfs attributes

```
/sys/class/davinci_dsplay/ch0/output
/sys/class/davinci_display/ch0/mode
```

For details of setting up and supported output and mode strings, please refer to the DaVinci Video Display Driver sysfs User's Guide.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265