Processor SDK Linux Automotive Display FAQ

Contents

Weston/Touch

1. Weston does not start

   To start weston needs a pointing device either a mouse or a touch screen. If you are using only HDMI or suspect that the touch screen is not working, please connect a mouse to the EVM.

2. `matrix-gui` is not responsive to touch but mouse works fine.

   It is possible that the touch screen calibration is incorrect. To redo the calibration, run

   ```
   root@dra7xx-evm:~# rm $WS_CALUDEV_FILE
   root@dra7xx-evm:~# sync
   ```

   reboot and do the calibration. This should get the touch working responsively.

3. LCD shows blank screen on first boot.

   Weston requires touch screen calibration before starting. You will notice a small cross hair on the LCD. Please touch the location of the cross hair until the touch screen is calibrated.

4. Touch screen is not working

   1. Check if the touch screen was probed correctly. If touch screen is probed correctly, you should see a node for touchscreen.

      ```
      root@dra7xx-evm:~# ls /dev/input
      by-path       event0        mice          mouse0        touchscreen0
      ```

      If you see the touchscreen node, run `evtest` as shown below, play with the touchscreen and check for touch events on the console output.

      ```
      root@dra7xx-evm:~# evtest /dev/input/touchscreen0
      Testing ... (interrupt to exit)
      Event: time 1493358175.185547, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 7
      Event: time 1493358175.185547, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 1381
      Event: time 1493358175.185547, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 225
      Event: time 1493358175.185547, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
      Event: time 1493358175.185547, type 3 (EV_ABS), code 0 (ABS_X), value 1381
      Event: time 1493358175.185547, type 3 (EV_ABS), code 1 (ABS_Y), value 225
      Event: time 1493358175.185547, ------------- SYN_REPORT ------------
      Event: time 1493358175.227251, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 1384
      Event: time 1493358175.227251, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 236
      ```

      If you do not see the touchscreen node under `/dev/input`, go to the rest of the steps.

   2. Check if the touch controller was detected correctly on the I2C bus. Assuming the touch controller was on i2c0, probe for I2C devices on the bus. Check that the touch controller is detected. The address of the touch controller can be determined from the device tree.

      ```
      root@dra7xx-evm:~# i2cdetect -y -r 0
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
      00:          -- -- -- -- -- -- -- -- -- -- -- -- UU --
      10: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
      20: UU UU -- -- -- -- -- UU -- -- -- -- -- -- -- --
      30: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
      40: UU -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
      50: -- 51 -- -- -- -- -- 57 UU UU UU 5b -- -- -- --
      60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
      70: -- -- -- -- -- -- -- --
      ```

      If touch controller was not detected, please check the ribbon cable connecting the EVM and the display for loose contact or cable breaks.

Weston Screen Rotation

Please refer to this response on the e2e forum.

https://e2e.ti.com/support/arm/automotive_processors/f/1020/p/629004/2321041#2321041

Clocking displays

Clocking LCD with HDMI PLL

If it is desired to clock one of the LCD outputs with the HDMI PLL instead of the Video PLL, the below patch can be used. Please note that HDMI needs to be disabled in the device tree and only one of the LCD outputs can use the HDMI PLL for output.

http://review.omapzoom.org/38396

This patch is only supported on K4.4.

The clocking architecture on K3.14 makes it cumbersome to support this same feature cleanly. As result, we do not support clocking LCD outputs with HDMI PLL on K3.14.

Printing clock setup

To understand the PLL setup at runtime, please use the script

```
# git clone git://git.ti.com/glsdk/util-scripts.git
# sh ./util-scripts/debug/dss_clockdumps.sh
====================DSS clock script==================
Dumps internal clocks and muxes of DSS

CTRL_CORE_DSS_PLL_CONTROL (0x4a002538) = 0x000002AB
video1 PLL :  Disabled
video2 PLL :  Disabled
HDMI   PLL :  Enabled
DSI1_A_CLK mux : DPLL HDMI
DSI1_B_CLK mux : DPLL video2
DSI1_C_CLK mux : DPLL Video1

DSS_CTRL (0x58000040) = 0x00000000
2: LCD1 clk switch :  DSS clk
3: LCD2 clk switch :  DSS clk
10: LCD3 clk switch :  DSS clk
1: func clk switch :  DSS clk
13: DPI1 output     :  HDMI

DSS_STATUS (0x5800005C) = 0x01408A81

DSI_CLK_CTRL (0x58004054) = 0x00000001

CM_DSS_CLKSTCTRL (0x4A009100) = 0x00040B03

CM_DSS_DSS_CLKCTRL (0x4A009120) = 0x00000702

=========================================================
Register dump for DPLL hdmi
|----------------------------|
| Address (hex) | Data (hex) |
|----------------------------|
| 0x58040200    | 0x00000018 |
| 0x58040204    | 0x00000003 |
| 0x58040208    | 0x00000000 |
| 0x5804020C    | 0x0004A40E |
| 0x58040210    | 0x00602004 |
| 0x58040214    | 0x00001800 |
| 0x58040218    | 0x00000000 |
| 0x5804021C    | 0x00000000 |
| 0x58040220    | 0x00040000 |
|----------------------------|
Details for DPLL hdmi
PLL status  :  Locked
M4 hsdiv(1) :  inactive
M5 hsdiv(2) :  inactive
M6 hsdiv(3) :  inactive
M7 hsdiv(4) :  inactive

PLL_REGM   =  594
PLL_REGN   =  7
M4 DIV     =  0
M6 DIV     =  0
M7 DIV     =  0
PLL_REGM2  =  1
PLL_REGM_F =  1
PLL_SD  =  6
HDMI_SSC_CONFIGURATION1(should be zero) 0x00000000
HDMI_SSC_CONFIGURATION2(should be zero) 0x00000000

Clock calculations (DPLL hdmi)
sysclk = 20000000
CLKOUT = sysclk * REGM / (REGM2 * (REGN + 1)) = 1485000000

=========================================================
Clock O/P of MUXes
DSI1_A_CLK :  1485000000
DSI1_B_CLK :  0
DSI1_C_CLK :  0

2: LCD1 clk :  192000000
3: LCD2 clk :  192000000
10: LCD3 clk :  192000000
1: func clk :  192000000
```

```
LCD1 logic clk(/ 1 ) :  192000000  pix clk(/ 2 ) :  96000000
LCD2 logic clk(/ 4 ) :  48000000  pix clk(/ 1 ) :  48000000
LCD3 logic clk(/ 4 ) :  48000000  pix clk(/ 1 ) :  48000000
```

Clocking two LCD's with same PLL

If you want to clock more displays than the available, one of the PLL's will need to be shared between the displays. Below is a set of patches on kernel 3.14 on J6 Eco (1 video PLL's and 1 HDMI PLL) for enabling both the LCD's from the same Video PLL.

Patches for Kernel 3.14

No.	URL	Headline
1	http://review.omapzoom.org/37626	omapdss: pll: fix writing M6 & M7 divs
2	http://review.omapzoom.org/37627	tmp:fix for VOUT1 + VOUT2 enablement to have LCD1 and LCD2
3	http://review.omapzoom.org/38278	HACK: Enable M4 and M6 dividers always.
4	http://review.omapzoom.org/38279	dra72: dts: enable vout2
5	http://review.omapzoom.org/38548	HACK: read VOUT2 PLL information from Video1 PLL

The below patch set offers similar functionality on Kernel 4.4.

Patches for Kernel 4.4

No.	URL	Headline
1	http://review.omapzoom.org/38519	drm/omap: track number of times dss pll is enabled
2	http://review.omapzoom.org/38520	drm/omap: add function to diff pll configuration
3	http://review.omapzoom.org/38521	drm/omap: HACK: allow use of a single PLL for two displays
4	http://review.omapzoom.org/38522	ARM: dts: dra7 evms: add dummy panels with AUO panel timings
5	http://review.omapzoom.org/38523	ARM: dts: dra7 : add dts for four displays
6	http://review.omapzoom.org/38524	ARM: dts: dra76 : add dts for four displays

Please ensure that you set the kernel boot argument `omapdrm.num_crtc` correctly when enabling multiple displays. The default kernel configuration is for 2 CRTC's. If you want to use 3 or 4 displays, please add `omapdrm.num_crtc=3` or `omapdrm.num_crtc=4` respectively to the kernel boot arguments.

Testing multiple LCD displays on the EVM

If you would like to test multiple identical LCD displays on the EVM, you can use the multiplexing option on VOUT1 to check the output of VOUT2 and VOUT3.

To do this,

1. Setup a fake panel(s) in the device tree and connect them to the appropriate device tree ports. See http://review.omapzoom.org/38522, http://review.omapzoom.org/38523
2. Set the clock sources so that all the panels have valid clocks. Either use the HDMI PLL for clocking one of the LCD's and modify kernel source such that same PLL clocks multiple displays using various PLL's.
3. Boot and check that the clock tree is as expected using the `dss_clockdumps.sh` script listed above.
4. Choose which output from VOUT1/2/3 goes on to LCD1 by modifying bits `DSS_CTRL[17:16]` address 0x58000040.

Testing overlays

You can test whether a particular overlay configuration works using `kmstest`. This tool is included in the target filesystem of Processor SDK - Linux Automotive 3.02. You can also build it from source from the link https://github.com/tomba/kmsxx/ . building requires the use of CMake and CMake tool chain file. An example toolchain file can be found below.

http://processors.wiki.ti.com/index.php/Processor_SDK_Linux_Automotive_Application_Notes#CMake

Below is an example running on a 1920x1200 display.

```
kmstest -c 0 \
-r 1920x1200 -f XR24 \
-p 0,0-640x480 -f 1920x1200 \
-p 100,100-640x480 -f 1280x960-NV12 \
-p 200,200-640x480
```

The breakdown of the command is described below.

1. Selects the first connector for the output.

   ```
   kmstest -c 0 \
   ```

2. Set the crtc format. As we are not specifying the frame buffer size, it is of the same size as the crtc.

   ```
   -r 1920x1200 -f XR24 \
   ```

3. Set an overlay starting from (0,0) of size 640x480. This buffer is generated by scaling a 1920x1200 buffer.

   ```
   -p 0,0-640x480 -f 1920x1200 \
   ```

4. Set an overlay starting from (100, 100) of size 640x480. This buffer is generated by scaling a 1280x960 NV12 format buffer. X and Y are both scaled down by 0.5.

```
-p 100,100-640x480 -f 1280x960-NV12 \
```

5. Set an overlay starting from (200, 200) of size 640x480. The framebuffer used is of size 640x480 as we did not specify a size.

```
-p 200,200-640x480
```

Here is an example of testing two displays simultaneously.

```
kmstest -c 0 \
-r 1920x1200 -f XR24 \
-p 0,0-640x480 -f 1920x1200 \
-c 1 \
-r 1920x1080 \
-p 100,100-640x480 -f 320x960-NV12
```

Debugging FPDLink integration

This section is written in the context of debugging FPDLink integration in the context of a display. Some of the steps are also applicable to in the context of FPDLink capture. In the following section, please interpret

1. FPDLink serializer as the local FPDLink device i.e. on the board with the Jacinto processor
2. FPDLink deserializer as the remote FPDLink device i.e. device connected to the display

The debug steps are as follows.

1. Ensure that the pinmux connections for the I2C lines and display data lines connecting to FPDLink serializer are setup correctly. Pinmux is usually done in MLO. You can verify pinmux settings using `omapconf`.

```
root@dra7xx-evm:~# omapconf read 0x4A003738
```

2. Verify that you are able to probe the FPDLink device. In case of the DRA74x EVM, FPDLink serializer DS90UH925Q is present on I2C2 bus with slave address of 0x1b. Please note the TRM and EVM I2C numbers are indexed from 1 while arguments to i2cdetect is indexed from 0.

```
root@dra7xx-evm:~# i2cdetect -r -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- UU UU UU 1b -- -- -- --
20: -- -- -- -- -- -- UU -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

3. Verify that you are able to read the configuration registers of the FPDLink local device i.e. serializer in case of display. Reading register 0 should return the 8 bit address of the serializer.

```
root@dra7xx-evm:~# i2cget -y 1 0x1b 0
0x36
```

4. Setup DSS to drive pixel clock from the desired output without any dependencies on backlight or board muxing.

5. Verify that the serializer is receiving the pixel clock. In case of DS90UH925Q, this can be determined by bit 2 of the "General Status" register (address 0xC).

```
root@dra7xx-evm:~# i2cget -y 1 0x1b 0xc
0x04
```

   If the serializer is not receiving pixel clock, please check the pinmuxing in the SOC and any muxes on the board.

6. Once the deserializer is connected via FPDLink cable, please check that communication is established between serializer and deserializer. This can be verified by reading bit 0 of the "General Status" register (address 0xC). The eight bit I2C address of the serializer should also be seen in DES ID register (address 0x6).

7. The FPDLink serializer and deserializers supported are listed in the file `Documentation/devicetree/bindings/video/fpd3-serdes.txt` in the kernel tree.

   a. Please check if your FPDLink device is supported. If not, support can be added by modifying `drivers/video/serdes/fpd3_serdes.c`.

   b. Each device has the initialization sequence defined in an array. Please review the initialization sequence against the desired configuration as per your board. Here is the initialization sequence for the 921 serializer.

   <source lang="c">static const unsigned int fpd3_921_ser_init[] = {

```
  /* Reset entire digital block except registers */
  FPD3_SER_RESET,     0x01,
  /* back chan en, auto ack WR, i2c passthrough, rising edge pclk */
  FPD3_SER_CONFIG1,   0x8b,
```

};</source>

{{

1. switchcategory:MultiCore=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **Processor SDK Linux Automotive Display FAQ** here.

Keystone=

- For technical support on MultiCore devices, please post your questions in the C6000 MultiCore Forum
- For questions related to the BIOS MultiCore SDK (MCSDK), please use the BIOS Forum

Please post only comments related to the article **Processor SDK Linux Automotive Display FAQ** here.

C2000=*For technical support on the C2000 please post your questions on The C2000 Forum. Please post only comments about the article* ***Processor SDK Linux Automotive Display FAQ*** *here.*

DaVinci=*For technical support on DaVincoplease post your questions on The DaVinci Forum. Please post only comments about the article* ***Processor SDK Linux Automotive Display FAQ*** *here.*

MSP430=*For technical support on MSP430 please post your questions on The MSP430 Forum. Please post only comments about the article* ***Processor SDK Linux Automotive Display FAQ*** *here.*

OMAP35x=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***Processor SDK Linux Automotive Display FAQ*** *here.*

OMAPL1=*For technical support on OMAP please post your questions on The OMAP Forum. Please post only comments about the article* ***Processor SDK Linux Automotive Display FAQ*** *here.*

MAVRK=*For technical support on MAVRK please post your questions on The MAVRK Toolbox Forum. Please post only comments about the article* ***Processor SDK Linux Automotive Display FAQ*** *here.*

For technical s please post yo questions at http://e2e.ti.co Please post o comments abo article ***Proces SDK Linux Automotive D FAQ*** *here.*

}}

Links

Amplifiers & Linear
Audio
Broadband RF/IF & Digital Radio
Clocks & Timers
Data Converters
DLP & MEMS
High-Reliability
Interface
Logic
Power Management
Processors

- ARM Processors
- Digital Signal Processors (DSP)
- Microcontrollers (MCU)
- OMAP Applications Processors

Switches & Multiplexers
Temperature Sensors & Control ICs
Wireless Connectivity