Application Report
SPRACD3–June 2018

Tools and Techniques to Root Case Failures in Video Capture Subsystem

Nikhil Devshatwar

ABSTRACT

This application report documents different tools and methods that are useful while debugging issues with the video capture subsystem on the Jacinto6™ (DRA7xx) family of system-on-chip (SoC). This can be used as a diagnostic test to root cause most common failures in the video capture use cases.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The DRA7xx family of SoCs have Video Input Port (VIP) modules used for video capture use cases. Some of these video ports are used for interfacing with external cameras. The following software components work together to get the video capture functional:

- Camera driver should configure the camera for desired hardware parameters
  - This typically happens via the Inter-Integrated Circuit (I2C) commands sent to the camera.
- Video signal needs to be routed correctly to the VIP video ports
  - Setup the board muxes and pinmux correctly
- VIP V4L2 driver should configure the video ports and VPDMA channels
- Camera V4L2 subdev driver should set the camera in streaming mode

Whenever there is a failure in the video capture, the symptom that is observed is very much the same from the application perspective. The V4L2 ioctl VIDIOC_DQBUF gets stuck and the application does not proceed. This document helps to determine the exact root cause by running a few diagnostic tests.

2 Supported Cameras

The processor software development kit (SDK) supports the following types of cameras.

2.1 Onboard LI Camera

Figure 1. OV10633

- OV10633 digital camera (in built ISP)
- UYVY 8-bit video up to 1280x800 discrete sync
- Programmable pclk, resolution, flip

This can be connected via a ribbon cable to the CPU board Leopard imaging connector.

No switch setting required

2.2 Vision Board Front Camera

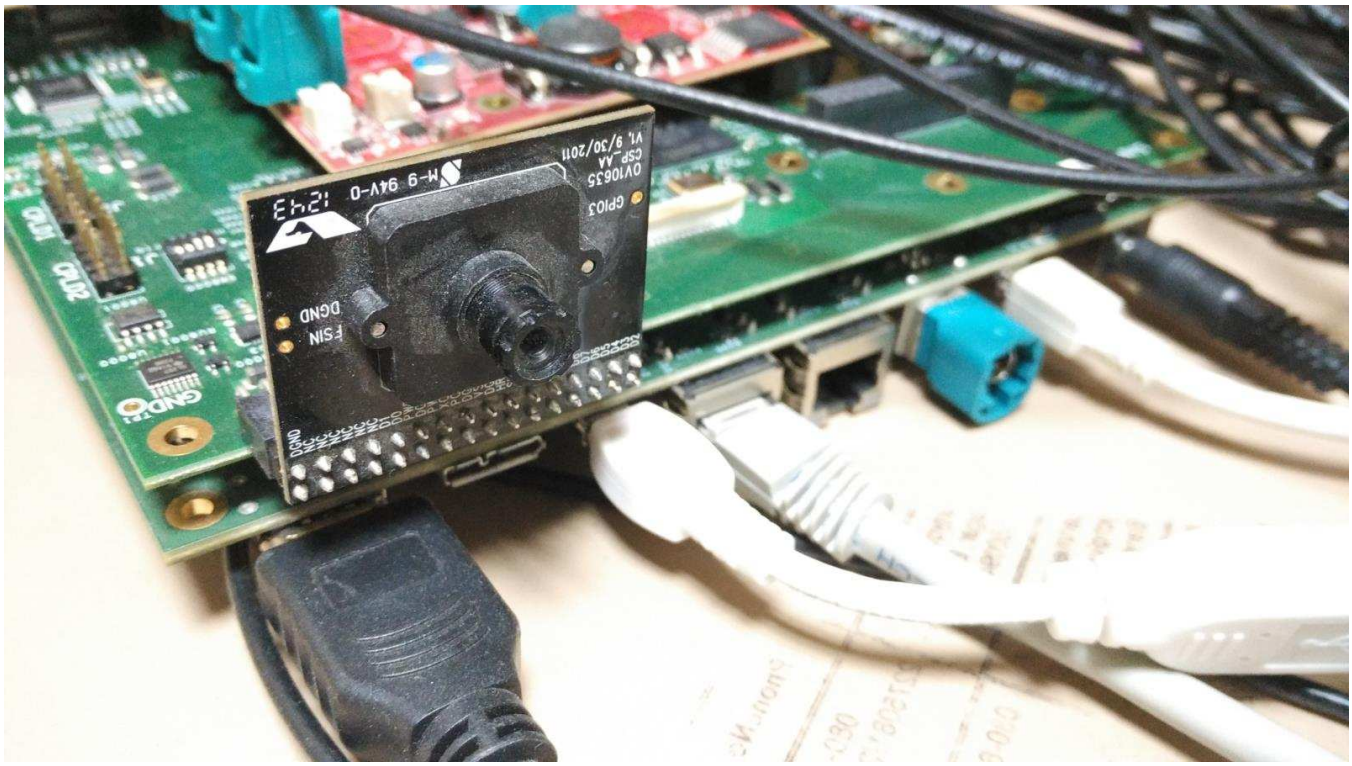

Figure 2. OV10635

- OV10635 digital camera (in built ISP)
- UYVY 8-bit video up to 1280x800 discrete sync
- Programmable pclk, resolution, flip

This can be connected using the Vision daughter card.

Vision board switch setting SW3[1-8] = 01010101

2.3 JAMR Board Analog Camera

Figure 3. analog camera via TVP5158

- Any analog camera with RF cable
- Texas Instruments analog video decoder
- UYVY 8-bit Embedded sync
- NTSC or PAL 720x244i

This can be connected via the JAMR application board. TVP5158 is used for the analog video decoder. Note that the TPV5158 can support **four** of these cameras and it generates multiplexed BT656 video. The multiplexing mode is configurable.

JAMR board switch setting SW2[1-2] = 01

2.4 Vision Board LVDS Camera

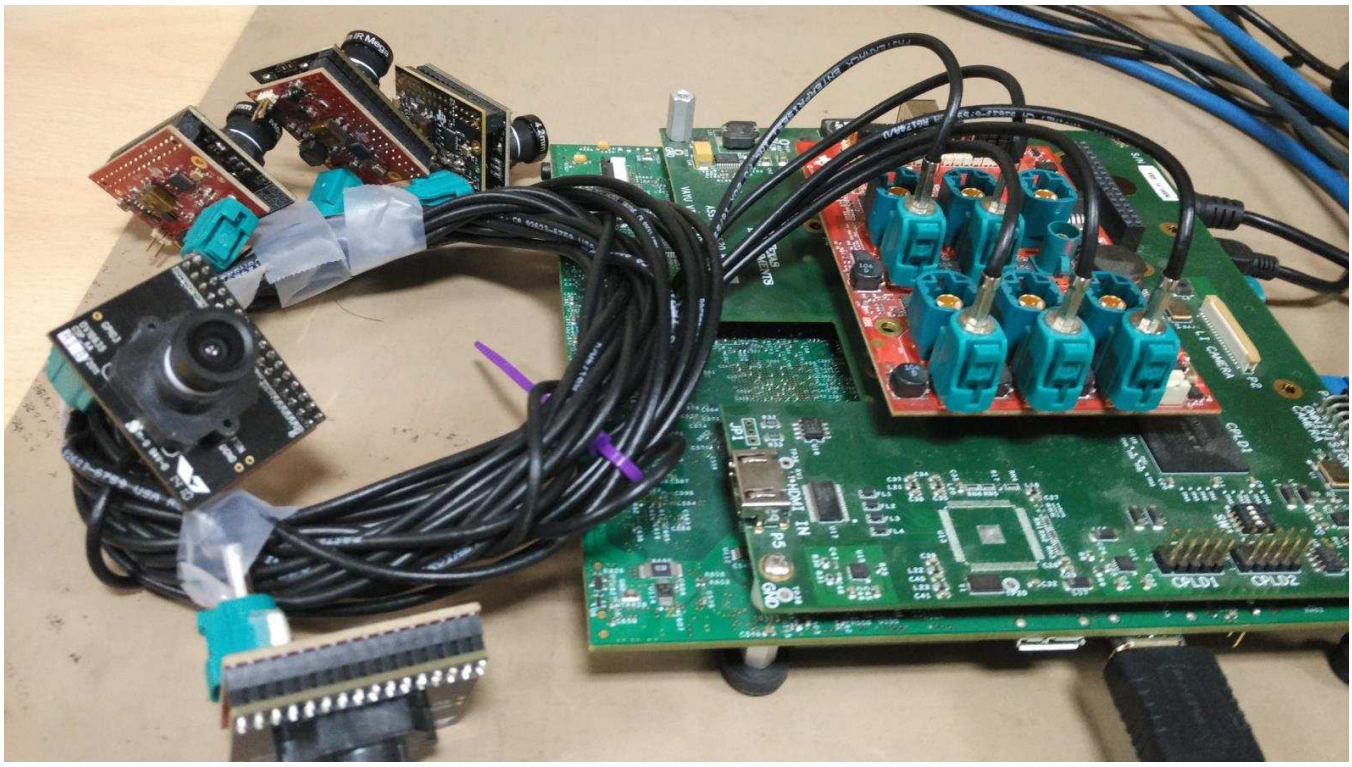

Figure 4. LVDS OV10635

- OV10635 camera with serializer/deserializer
- 1280x720 UYVY 8-bit discrete sync
- DS90UB913 and DS90UB914 12-bit video serializer/deserializer

This can be connected using the Vision daughter card along with the multi deserializer board setup. Serdes needs to pair with each camera.

Vision board switch setting SW3[1-8] = 00100101

The following diagnostic checks can be performed to root cause failures.

3 Failure to Launch Video Capture Application

The first thing to look for is whether the video devices are created or not. Check the bootlog for prints in the kernel bootlog. Run the following on the target:

```
dmesg | grep ov1063x
dmesg | grep tvp
dmesg | grep video
```

Depending on the camera connected, the following prints can confirm the probe being successful, (see Table 1).

Table 1. Kernel Log Results

Kernel Log	Result
ov1063x 1-0037: ov1063x Product ID a6 Manufacturer ID 33	Onboard camera probe success
ov1063x 1-0030: ov1063x Product ID a6 Manufacturer ID 35	Vision Front camera probe success
tvp5158 4-0058: Camera sensor driver registered	TVP decoder probe success
ov1063x X-00XX: Failed writing register 0x0103!	Camera not connected

4 Failure to Start Video Capture

When the capture application is launched, it is expected to start video capture and display the frames onto the display. Sometimes, no video is displayed on the screen. To identify this being an issue with capture, a simple test can be done. Each VIP slice has a dedicated interrupt line. If the capture is successful, the interrupt count should increase periodically. The following are some of the common failure scenarios failing to launch the capture application.

```
cat /proc/interrupts | grep vip

362:        941          0       GIC 102  vip1-s0
363:        183          0       GIC 101  vip1-s1
364:        241          0       GIC 100  vip2-s0
365:          0          0       GIC  99  vip2-s1
366:         46          0       GIC  98  vip3-s0
367:          2          0       GIC  97  vip3-s1

#    this   ^^^^    number indicates the number of interrupts.
```

If the interrupt values for the desired video port do not change, while the application is running, then the video capture is failing.

NOTE: Due to the VPDMA FIFO handling, you may see two interrupts (= size of VPDMA FIFO) getting fired; do not count them. The first two interrupts indicate VPDMA processing the descriptors. It is only the **third interrupt after capture start** that indicates a frame completion.

NOTE: This behavior is changed from Processor SDK 3.0 onwards where the **first interrupt after capture start** indicates the frame completion.

In the above example, you can conclude that * capture from Vin1, Vin2, Vin3, Vin5 is working fine. * Vin4(vip2-s1) capture was never attempted. * Vin6(vip3-s1) capture is failing (Only FIFO IRQ, no frame completion IRQ)

Also note that the IRQs are shared for different ports of the same slice. This means, vip1-s0 line will carry interrupts from both vin1a and vin1b. This test can be used when only one of the port is in use.

4.1 Parser is not Able to Detect the Video

Most of the time, external factors cause this failure. For a new board bringup, this is the most common issue.

As soon as the video port detects the sync signals, parser updates the detected video size in the PARSER_SIZE register. This is useful for finding out whether the video signals are getting to the VIP port or not. Note that, the parser size is calculated only based on the relative toggling of pclk, hsync, vsync. Also, the size includes any blanking data available in the stream.

Table 2 lists important parser registers for different video ports. If the video port you are suing is not listed below, see the device-specific technical reference manual (TRM).

Table 2. Parser Registers for Video Ports

Video Port	Parser Size Register	Parser Config Register
vin1a	0x48975530	0x48975504
vin1b	0x48975570	0x4897550C
vin2a	0x48975A30	0x48975A04
vin2b	0x48975A70	0x48975A0C
vin3a	0x48995530	0x48995504
vin3b	0x48995570	0x4899550C
vin4a	0x48995A30	0x48995A04
vin4b	0x48995A70	0x48995A0C
vin5a	0x489B5530	0x489B5504
vin6a	0x489B5A30	0x489B5A0C

- If the parser config register shows 0x0 values, it indicates either the capture application is not running or the software is using the wrong video ports.
- If the parser size register shows 0x0 values, it indicate the parser is not able to detect the video.

The following subsections describe various root causes for this type of failure.

4.1.1 Invalid Parser Configuration

Depending on the camera used, certain parameters of the video port need to be configured correctly. Device tree definition (endpoint nodes) is used for specifying these parameters.

Usecase	Required Parameters
Parallel port	Bus width (8/16 bit for YUV, 24 bit for RGB)
Descrete sync	hsync, vsync, pclk polarities
Embedded sync	Multiplexing method, channel numbers

To check whether the correct parameters are being passed or not, procfs can be used for checking values of some of the properties on the target.

Run the following commands on the target:

```
cat /proc/device-tree/ocp/i2c@480720000/ov10635@37/compatible
# Replace the device node path ^^^^^^^^^^^^^^^^ as appropriate

hexdump -b /proc/device-tree/ocp/i2c@480720000/ov10635@37/port/endpoint@0/pclk-sample

hexdump -b /proc/device-tree/ocp/i2c@480720000/ov10635@37/port/endpoint@0/bus-width

hexdump -b /proc/device-tree/ocp/i2c@480720000/ov10635@37/port/endpoint@0/channels
```

NOTE: Some of the integer properties are not printable in ASCII format. Using hexdump gives readability to read integer values from device tree.

4.1.2 Incorrect Board mux or pinmux Configuration

On the dra7xx family of SoCs, all the pads can be probed (at very small sampling rate) on the target by using the GPIO datain registers.

To generate the probe scripts, see the **iodelay-config** host side utility.

IO-delay config utility

This script should be run to create a simple bash script that can be run on the target to probe the pins being used for a specific video interface. The following is an example of the probe script used for vin1a interface:

```
./vin1a-probe.sh
Probing VIN1A signals
    CLK0   HSYNC0  VSYNC0      D0      D1      D2      D3      D4      D5      D6      D7
       1       0       0       0       1       0       0       1       0       1       0
       1       0       0       1       0       0       1       1       0       0       0
       1       1       0       0       1       0       1       0       1       0       0
       1       0       0       1       0       0       1       1       0       0       1
```

The example above shows a live signal probe of the vin1a interface on DRA75x board. Note that this test is only to indicate availability of the signal. No conclusion regarding timing should be derived as the values are sampled at very low rate. Also, for most cases, the vsync signal will not toggle from 0 to 1 because it is only active for a short duration. Failure to identify vsync signal is expected.

4.1.3 Camera is not Started, pclk, Syncs are Dead

This is a root cause where the camera board is not generating video signals in the desired format. Subdevice s_stream op is supposed to perform all the I2C transactions to indicate sensor to start streaming. Failing to get the pixel clock at this time indicates some issue in the camera configuration. Most cameras have a power pin driver by one of the GPIO, make sure that the subdev driver requests for this GPIO

4.1.4 Serializer/Deserializer Configuration Issues

When a camera is connected over a serializer/deserializer (LVDS) pair, there are more items to be investigated. Even though the camera maybe functional, if the Serdes link is not setup properly, SoC will not get any video from the remote camera. Serdes configuration varies from different part numbers. Some common tests help identify problems in the Serdes configuration. As explained above, each deserializer is a remote I2C bus, so basic I2C tools can be used to check the accessibility.

The following utility script runs a bunch of accessibility tests to identify common issues with Serdes — vip_diagnostic.sh:

```
./vip_diagnostic.sh vin1a

====================VIP diagnostic script==================
Basic tests to debug capture issues quickly

VIP 1 Slice 0 Port A === LVDS cam1

=>           Check if the DE-SERIALIZER is accessible          PASS
=>               Check if the SERIALIZER is connected          PASS
=>              Check if the SERIALIZER is accessible          PASS
=>               Check if the CAMERA clock is present          PASS
=>                  Check if the CAMERA is accessible          PASS
=>                  Check if the parser is configured          PASS
=>      Check if the port is detecting the frame size          FAIL
```

The example above shows the test run of a LVDS camera (cam1) connected to the vin1a interface of the DRA74x board with the Vision application board. Each test is an indicative of accessibility of devices from local bus to the remote camera. Use the right video port name depending on the interface being used for the camera.

4.2 Capture Stuck at Runtime

There are cases where even if the capture started working, it may get stalled at some point. This might be due to some operation like (start/stop of same/other stream) or just testing for long run.

Run some of the following diagnostic tests to root cause such failures.

4.2.1 VPDMA List Status

VPDMA is the hardware block responsible for the DMA of the video frames. There is only one VPDMA block for one VIP instance and DMAs from all slices/ports/channels go through this same VPDMA. There are different channels of VPDMA used: one for each stream.

The VPDMA descriptor dump might be able to describe the failure signature. The following cmds demonstrate how to get the VPDMA descriptor dump.

Depending on the VIP instance used, the base address will be different.

Instance	VPDMA Base Address
VIP1	0x4897d000
VIP2	0x4899d000
VIP3	0x489bd000

```
VPDMA list address = VPDMA base + 0x04
VPDMA list attributes = VPDMA base + 0x08
VPDMA list status = VPDMA base + 0x0c
```

The following method can be used to get the VPDMA dump

```
#Read VPDMA desc start address, save this in addr
addr=`omapconf read 0x4897d004 | grep -v omapconf | grep -v grep`
#Read VPDMA descriptor dump
omapconf dump 0x$addr 0x`printf %x $(( 0x$addr + 0x40))`
```

NOTE: You can use the vip_diagnostics.sh to get the VPDMA descriptor dump as well.

The VPDMA list status indicates if the DMA is ongoing(1) or done(0). A busy VPDMA list indicates the VPDMA is waiting for data to be sent from the VIP parser but somehow is not getting any.

4.2.2 VPDMA Channel Stalled

VPDMA hardware allows a query of the state of a specific VPDMA list using the following sequence:

```
#Write 0xdead000 NUM into the VPDMA pid register
#e.g. for list3 on VPDMA of VIP2
omapconf write 0x4899d000 0xdead0003
```

The output of the above command can be interpreted as shown in Table 3.

Table 3. VPDMA PID Value Status

VPDMA PID Value	List Status
0x1FF	List inactive
0x1FE	List Active and free running
0x1FD	Illegal Control Descriptor
0x1FC	Configuration Descriptor Operation
0x1FB	Regular List operation
0x1FA	List Queue Empty Operation
0x1FD - 0x1F0	List Active, debug label
0x1EF - 0x000	List blocked by channel

5 Summary

This document introduces some tools and techniques that are helpful in root causing the most common issues faced by customers. The symptoms for most of the failures look the same from the application perspective. When you run these diagnostic tests, you get more details on the failure. Most of the diagnostic test cases also point to the exact cause of the failure. This document may serve as a procedural guide to debugging issues with video capture subsystems.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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