Application Report
SPRACG3–August 2018

The Implementation of YUV422 Output for SRV

Marvin Liang, Brijesh Jadav and Subhajit Paul

ABSTRACT

From a design perspective, the display subsystem (DSS) in the Jacinto™/TDA family of devices only supports RGB or non-standard embedded sync YUV422 output. This technical limitation creates a big problem when the car OEM integrates the ADAS system to an infotainment system, especially when replacing the rear view camera with the surround view function. This application report introduces a solution that outputs discrete YUV422 for a SRV system based on both VISION SDK and PSDKLA.

Project collateral and source code discussed in this document can be downloaded from the following URL: http://www.ti.com/lit/zip/spracg3.

Contents

List of Figures

Trademarks

Jacinto is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

1 Requirement for Supporting YUV422 Output

As shown in Figure 1, currently most cars have RVC function integrated, it connected to the infotainment system through FPD LINK/LVDS, the video format for RVC camera is YUV422 and in most cases, it use 8 bit interface. Now more OEMs plan to upgrade the RVC to 360 surround view function, or the RVC remain for low end car, SRV remain for medium/high end car. Because most SRV products which based on TI Jacinto/TDA family devices are only designed for RGB output and TI FPD LINK product always is a serializer/deserializer pair for specific video format conversion, for example UB933/UB934 for YUV422, UB921/UB924 for RGB, the infotainment system have to integrate two different deserializers for support both RVC and SRV function.

If SRV can output YUV422 as RVC camera, infotainment system will only need one YUV422 deserializers to support both RVC and SRV function.

It takes simplicity for the infotainment system design and flexibility for system deploying, also the cost benefit.

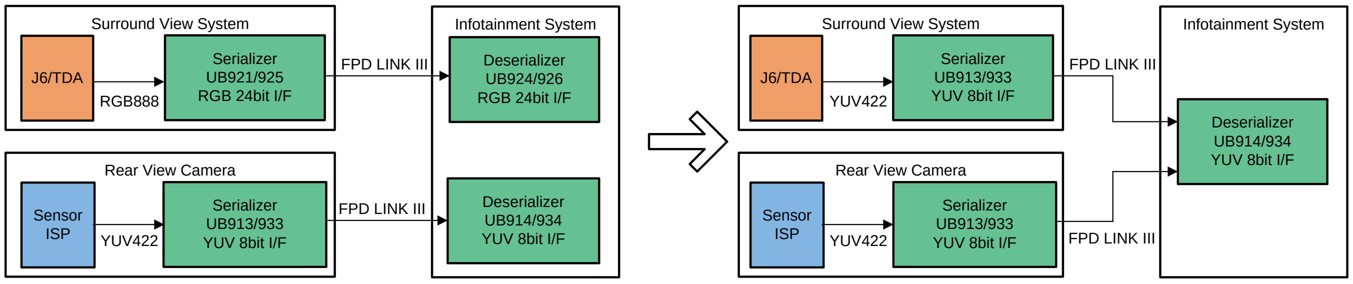

Surround View System
J6/TDA
RGB888
Serializer UB921/925 RGB 24bit I/F
FPD LINK III
Infotainment System
Deserializer UB924/926 RGB 24bit I/F
Rear View Camera
Sensor ISP
YUV422
Serializer UB913/933 YUV 8bit I/F
FPD LINK III
Deserializer UB914/934 YUV 8bit I/F
Surround View System
J6/TDA
YUV422
Serializer UB913/933 YUV 8bit I/F
FPD LINK III
Infotainment System
Deserializer UB914/934 YUV 8bit I/F
Rear View Camera
Sensor ISP
YUV422
Serializer UB913/933 YUV 8bit I/F
FPD LINK III

Figure 1. Benefit for Supporting YUV422 Output in SRV System

2 YUV422 Format and Output Timing

According to the TRM for DSS, DSS support output below format:

1. Embedded sync output interface for YUV422 format
2. Supports BT656 and BT1120 standards
3. Supports YUV422 format with 8bit color component
4. Outputs 8bit YUV422 over 10bit interface with valid pixel values on D[9:2] data lines
5. Discrete sync output interface for various bit depth of RGB format (RGB888, RGB565, RGB444)

2.1 Emded Sync Output

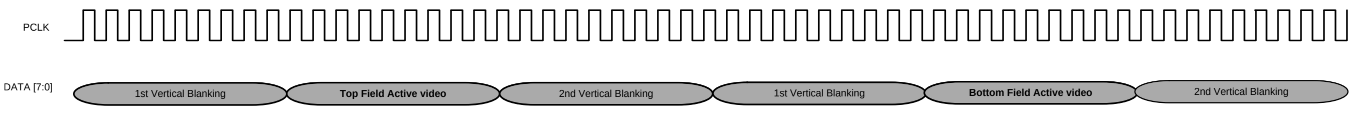
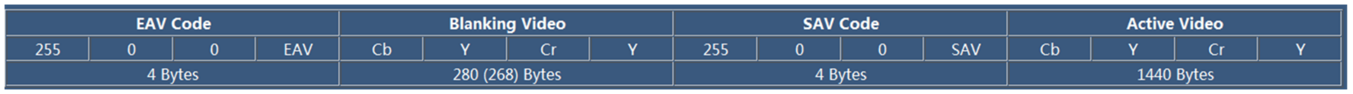


Figure 2. BT.656 Frame Timing



EAV Code				Blanking Video				SAV Code				Active Video			
255	0	0	EAV	Cb	Y	Cr	Y	255	0	0	SAV	Cb	Y	Cr	Y
4 Bytes				280 (268) Bytes				4 Bytes				1440 Bytes			

Figure 3. One Line Active Video Format for BT.656

For embedded sync output video format, only need PCLK and data lines. The synchronization signals are already packed into the video content as shown in Figure 2 and Figure 3. The receive side does not need HSYNC and VSYNC, but the video can be decoded with the packed vertical blanking, start of active video (SAV), and end of active video (EAV).

But there is a limitation for TI Jacinto/TDA family devices, as below CAUTION part in TRM:

CAUTION

DISPC supports maximum 256 bytes of horizontal blanking when the LCD outputs are configured in BT modes (bitfield HSW of DISPC_TIMING_H1/2/3 registers is 8 bits wide). This is not compliant with BT.656 and BT.1120 standards, both of which require higher blanking periods. If more than 256 bytes of horizontal blanking are required by the application, then the RGB mode must be used for the LCD outputs.

The Blanking Video definition in ITU-R-BT.656 (as shown in Figure 3) is 280 byte for PAL, 268 byte for NTSC, so it is very difficult to find a BT.656 decoder that can support the nonstandard BT.565 signals (blanking video 256 byte) from TI Jacinto/TDA family devices.

2.2 Discrete Sync Output

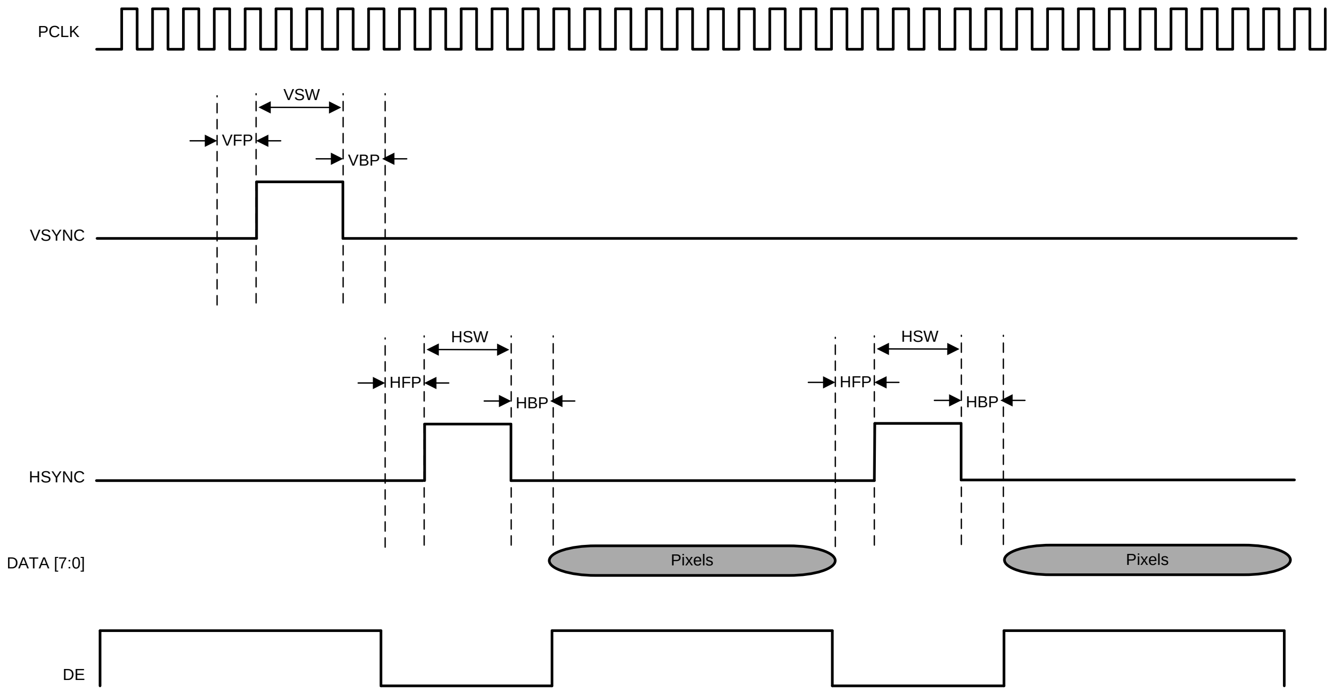


Figure 4. Discrete Sync Output Timing

DSS output a discrete video in Jacinto/TDA family devices shown as Figure 4:

1. Supports HSYNC, VSYNC, DE signals
2. Outputs 8bit data over 8bit interface with valid pixel on D[7:0] data lines
3. Supports multi cycle output format (TDM), where each pixel can be sent out over multiple clock cycle

VSYNC: Vertical Synchronization signal, it toggles after all the lines in a frame are transmitted to the LCD panel and a programmable number of line clock cycles has elapsed at the beginning and end of each frame. There are three programmable variables to control VSYNC timing, VFP, VBP and VSW.

HSYNC: Horizontal synchronization Signal, it toggles after all pixels in a line are transmitted to the LCD panel and a programmable number of pixel clock has elapsed at the beginning and end of each line. There are three programmable variables to control VSYNC timing, HFP, HBP and HSW.

DE: Data Enable Signal, it acts as an output-enable signal to indicate when data must be latched using the pixel clock. DE timing depends on the HFP, HBP and HSW configuration

DSS adds one to the configured value for the field HFP, HBP and HSW, so for these fields, the minimum period is one clock cycle. If these fields are configured with value 0, DSS outputs single clock cycle for these fields.

All three sync signal can be inverted in the DSS.

Because of the limitation of BT.656, the implementation of this application note is targeted to discrete sync YUV422.

3 Discrete Sync YUV422 Output Concept

3.1 *The Function Modules Bypass in VID Pipeline*

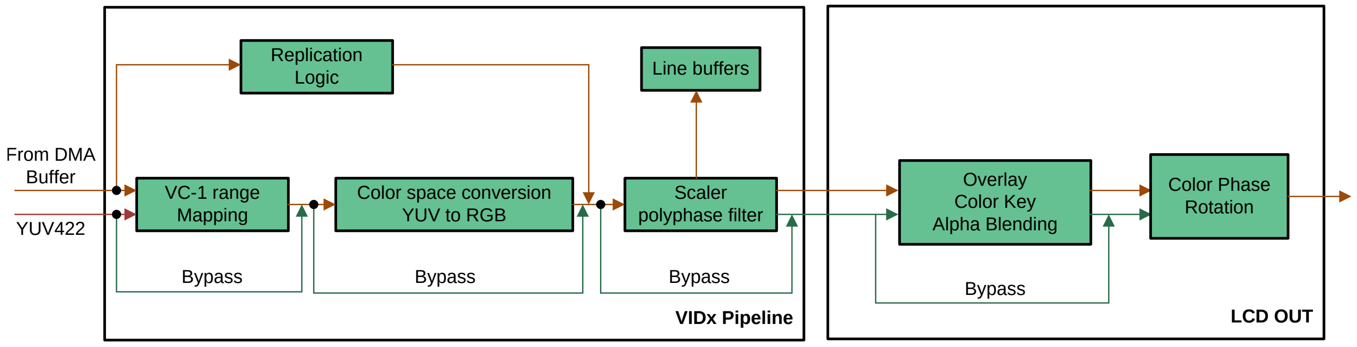


Figure 5. Bit Exact Output YUV422 From VIDx Pipeline

Since our DSS CSC submodule cannot convert RGB to YUV, the basic concept to get YUV422 output is that input YUV422 format to a specific video pipeline and get bit exact output.

For providing the overlay feature, it requires the input video format as RGB, because only RGB can have alpha characteristic. So the video pipeline always converts the non-RGB input to RGB format internally. When input video is YUV422 format and configure the format to YUV422, video pipeline will convert it. For voiding this default hardware behavior, the hardware needs to be cheated to deal the YUV422 as RGB. Since both YUV422 and RGB565 are 16 bit data, cheat hardware deal YUV422 as RGB565 is possible.

Each of the processing blocks present in the video pipelines has a bypass path (see Figure 5), which can be selected to send the data unaltered by the video pipeline. Input data format of the video pipeline should be configured as RGB16-565. VC-1 range mapping and CSC blocks of the video pipelines will be bypassed by default for this format. Input and output width and height of the video pipeline should be configured to same values to bypass the scaling block. With the above configuration the data coming to video pipeline from DMA engine will reach the overlay unaltered.

Register Configuration:

```
DISPC_VID1_ATTRIBUTES.FORMAT = 0x6; RGB16-565
DISPC_VID1_ATTRIBUTES.RESIZEENABLE = 0x0; Disable both horizontal and vertical resize
processing
DISPC_VID1_SIZE.SIZEX == DISPC_VID1_PICTURE_SIZE.MEMSIZEX
DISPC_VID1_SIZE.SIZEY == DISPC_VID1_PICTURE_SIZE.MEMSIZEY
DISPC_CONFIG1.TCKLCDENABLE = 0; Disable the transparency color key for the LCD
DISPC_CONFIG1.CPR = 0; Color Phase Rotation Disabled
}
```

3.2 TDM for Transferring YUV422

Since RGB565 and YUV422 are both 16 bit per pixel data, for transferring a pixel through 8-bit interface, it can be divide to dual 8 bit, each one can be transferred by one cycle. Figure 6 shows the TDM timing to output YUV22 pixel.

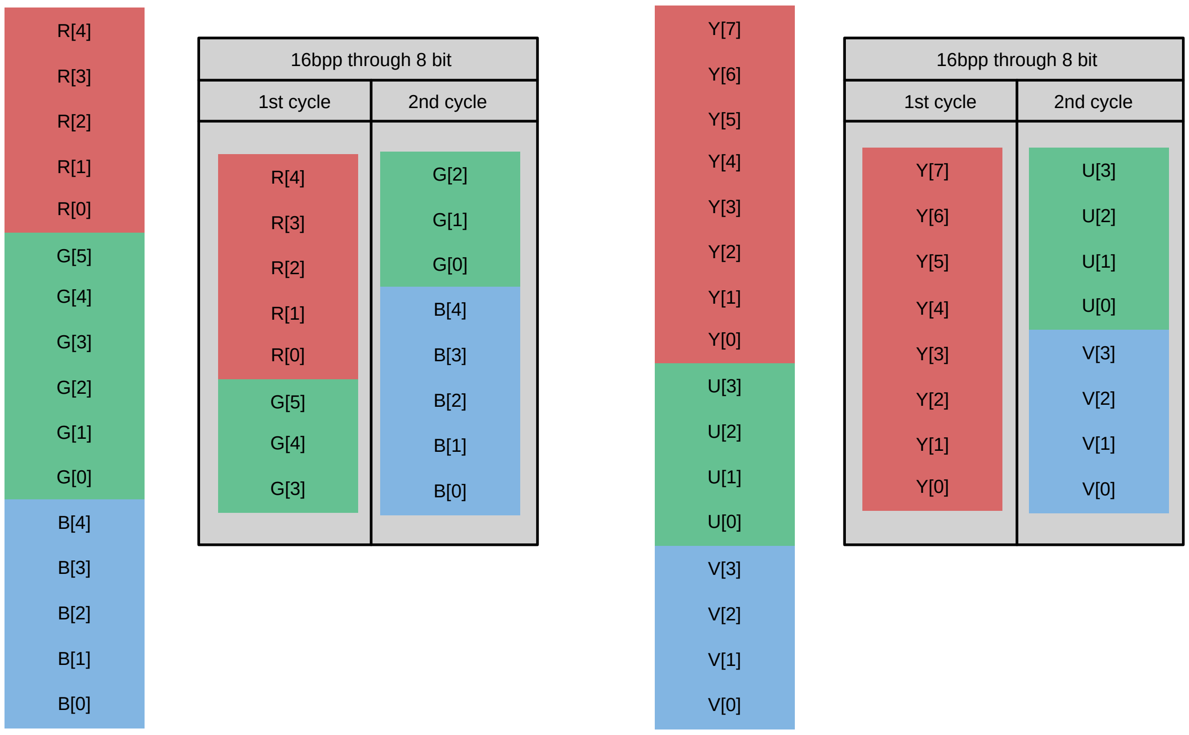


Figure 6. YUV422 Output Through TDM

Register Configuration:

```
DISPC_VP1_CONTROL.TDMENABLE = 0x1: TDM enabled
DISPC_VP1_CONTROL. TDMPARALLELMODE = 0x0: 8-bit parallel output interface selected
DISPC_VP1_CONTROL. TDMCYCLEFORMAT = 0x2: 2 cycles for 1 pixel
DISPC_DATA1_CYCLE1 = 0x8
DISPC_DATA1_CYCLE2 = 0x8
DISPC_DATA1_CYCLE3 = 0x0
```

4 Migration From RGB888 to YUV422 Output for SRV Use Case

Considering a typical SRV use case, Figure 7 provides a system level overview that how to migrate from RGB888 output to YUV422 output. This use case example is based on J6 Entry/DRA71x, which does not have the VOUT1 interface.

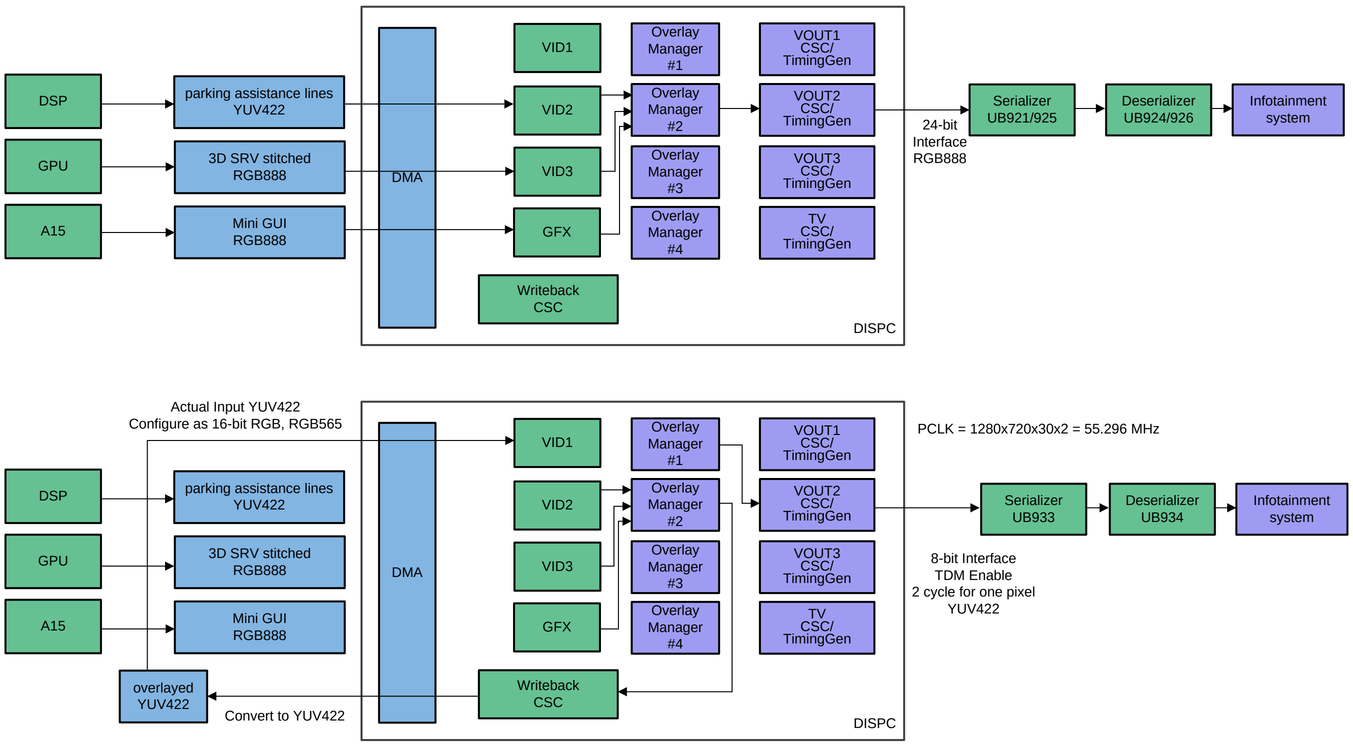


Figure 7. YUV422 Output for SRV Use Case

DSP algorithm output YUV422 frames for parking assistance lines bind to VID2 pipeline

GPU output stitched 3D SRV RGB888 frame bind to VID3 pipeline

A15/MiniGUI output the RGB888 UI frame bind to GFX pipeline

After the overlay for above three frames, it can output RGB888 through 24 bit interface directly. For outputting YUV422, it has to be routed to writeback pipeline and convert the RGB888 to YUV422 in the memory. Then an application should schedule the new YUV422 frame as input to VID1 pipeline. The VID1 pipeline has been already configured as YUV422 bit exact output that are described Section 3.1 and Section 3.2.

4.1 *YUV422 output migration for SRV on VISIDONSDK*

The YUV422 output migration for SRV from RGB888 output on VISIONSDK is not so complex. Most modifications are made for display control link, only need add one more stage, the writeback capture link. Figure 8 shows the VISIONSDK link for the SRV use case with YV422 output.

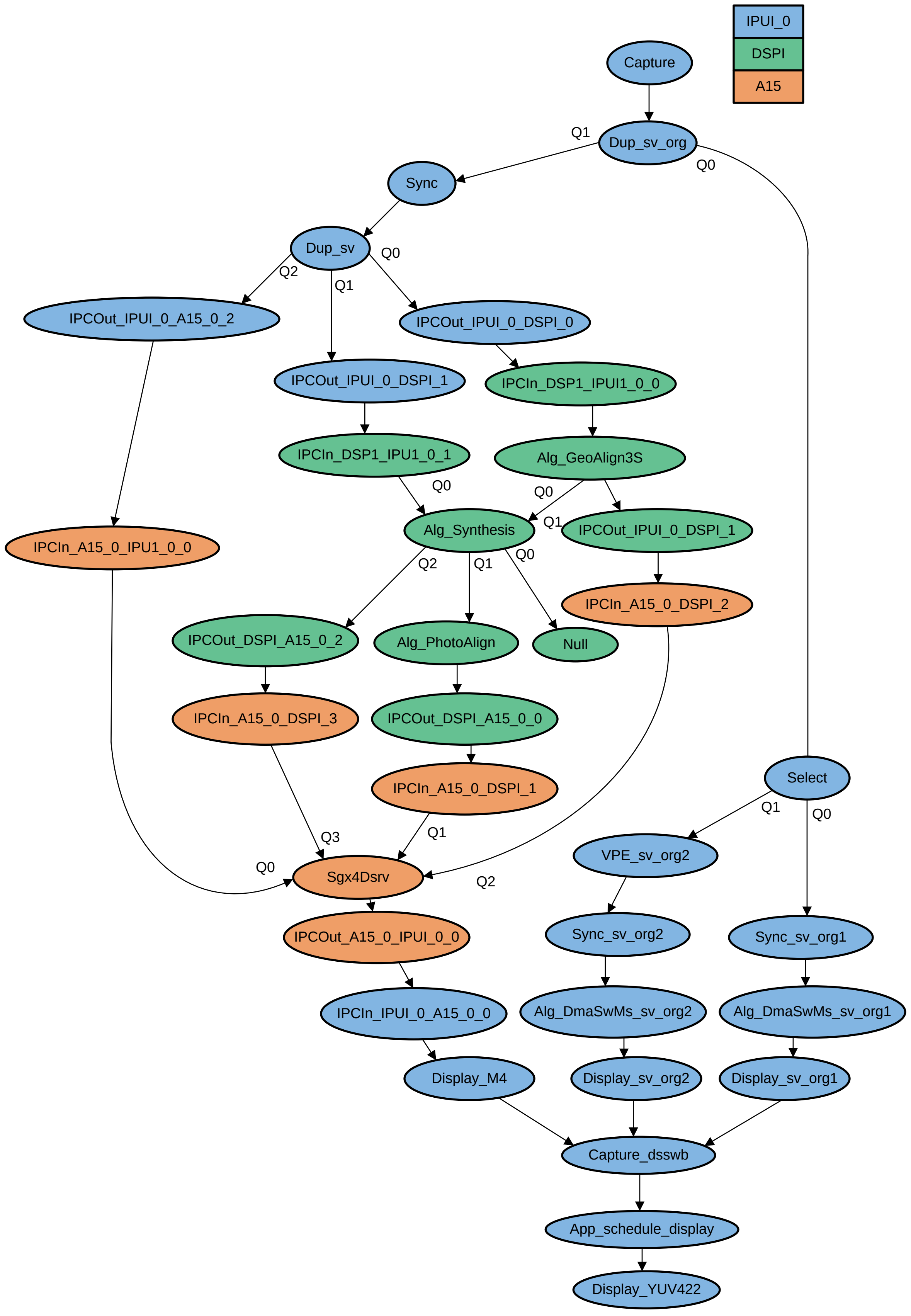


Figure 8. YUV422 Output Chain for SRV

4.2 YUV422 Output Migration for SRV on PSDKLA

When implement the YUV422 output in PSDKLA, it is more difficult than VISION SDK since Linux have the display driver framework as DRM. The modification cannot be made straight forward like VISIONSDK M4 display link/driver, it has to be followed the DRM methodology.

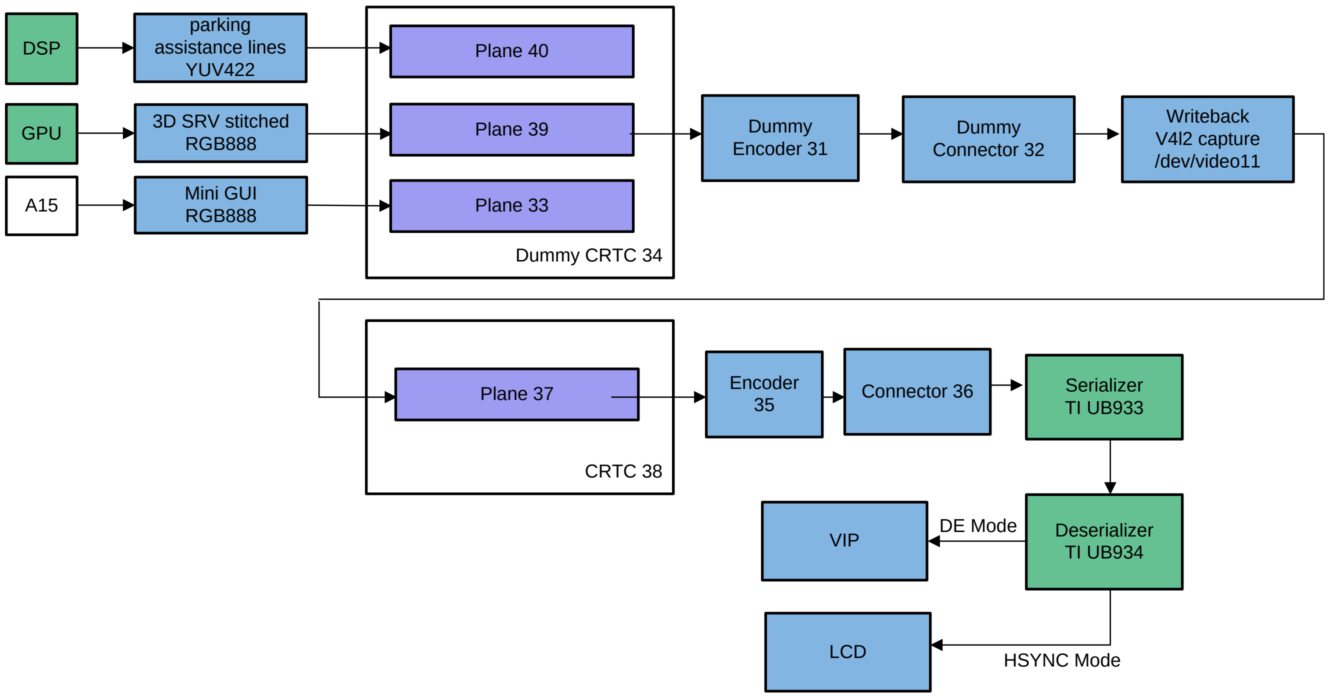


Figure 9. YUV422 Output Implementation for SRV in PSDKLA

Figure 9 shows the YUV422 output implementation for SRV with Linux DRM framework:

1. DSP algorithm output YUV422 frames for parking assistance lines bind to CRTC34-Plane40 (VID2 pipeline)
2. GPU output stitched 3D SRV RGB888 frame bind to CRTC34-Plane39(VID3 pipeline)
3. A15/MinuGUI output the RGB888 UI frame bind to CRTC34-Plane33 (GFX pipeline)

   Note: The CRTC34 is a dummy CRTC which only have dummy encoder and connect, no actual video output to outside the DSS.

1. A Linux user mode program should take the buffer from the DSS writeback capture device (/dev/video11) by standard V4L2 driver, also configure the output format of /dev/video11 as YUV422.
2. The captured YUV422 frame can be bind to CRTC38-Plane37 (VID1 pipeline)

The corresponding encoder/connector for CRTC38 output the YUV422

4.3 Verification for YUV422 Output

As shown in Figure 9, the YUV422 output can be verified by two ways:

1. Loopback the serializer of DSS output to the deserializer of J6 Entry's VIP. For this case, prefer to use DE signal instead of using HSYNC signal from DSS. VIP should capture the excellent YUV422 video without requiring the cropping of data.
2. Connect the serializer of DSS output to the deserializer of a LCD which supported YUV422 input. For this case, prefer to use HSYNC signal directly from DSS, because most LCD are requiring the Hblanking timing (HFP/HSW/HBP).