Robust Rear-View Camera (RVC)

ABSTRACT

This application report is a robust, rear-view camera use case designed for the Android® high-level operating system (HLOS), in addition to the Vision SDK release, which runs on the Cortex®-M4 IPU for robustness.

This application report provides the procedure to customize and adapt the default configuration on the Robust RVC release.

Contents

List of Figures

Trademarks

1 Introduction

Table 1 lists the acronyms, abbreviations, and definitions.

Table 1. Acronyms, Abbreviations, and Definitions

Acronym / Abbreviation	Definition
BIOS	Basic input and output system
CRC	Cyclic redundancy check
DTS	Device tree used in Android kernel
DSP	Digital signal processor
DSS	Display subsystem
DSS WB	Display subsystem write back
GPIO	General purpose input/output
HDMI	High definition multimedia interface
HS	High security
HLOS	High-level operating system
IVA	Image video accelerator
IPU	Image processing unit
PDK	Processor development kit
PROCESSOR_SDK_VISION_03_xx	Vision SDK 3.x package
RVC	Rear view camera
VID	Video input display
VIP	Video input port
VPE	Video processing engine
VPDMA	Video port direct memory access

This application note provides guidelines to modify the Robust-RVC use case for Jacinto™ Infotainment processors. The application report explains the steps to enable the following features for the Robust RVC solution on the Android operating system (OS).

- New camera sensor
- DSS and configuring display panel
- HDMI
- Robust RVC on different cores, such as IPU1 and DSP1
- Robust RVC diagnostics
- Firewall configuration on HS devices
- Frame-freeze detect
- Message exchange between A15 and IPU

2 Robust RVC Overview

The Robust RVC solution provides the following functionalities:

- DRA7xx IPU (Cortex-M4)-based handling of Robust RVC functions:
  - Provides isolation from the HLOS running on the MPU (Cortex-A15)
  - Provides solution for early camera after system power-on, IPU can run Robust RVC while the HLOS on the MPU is still booting.
- Bootloader design that loads the IPU code and data to SDRAM, and starts the IPU before jumping to the HLOS kernel boot – enabling early RVC < 2 seconds
- Initialize VIP, VPDMA, VPE, and DSS (all from the IPU), and get the Robust RVC frames quickly displayed using a DSS VID pipe after boot.

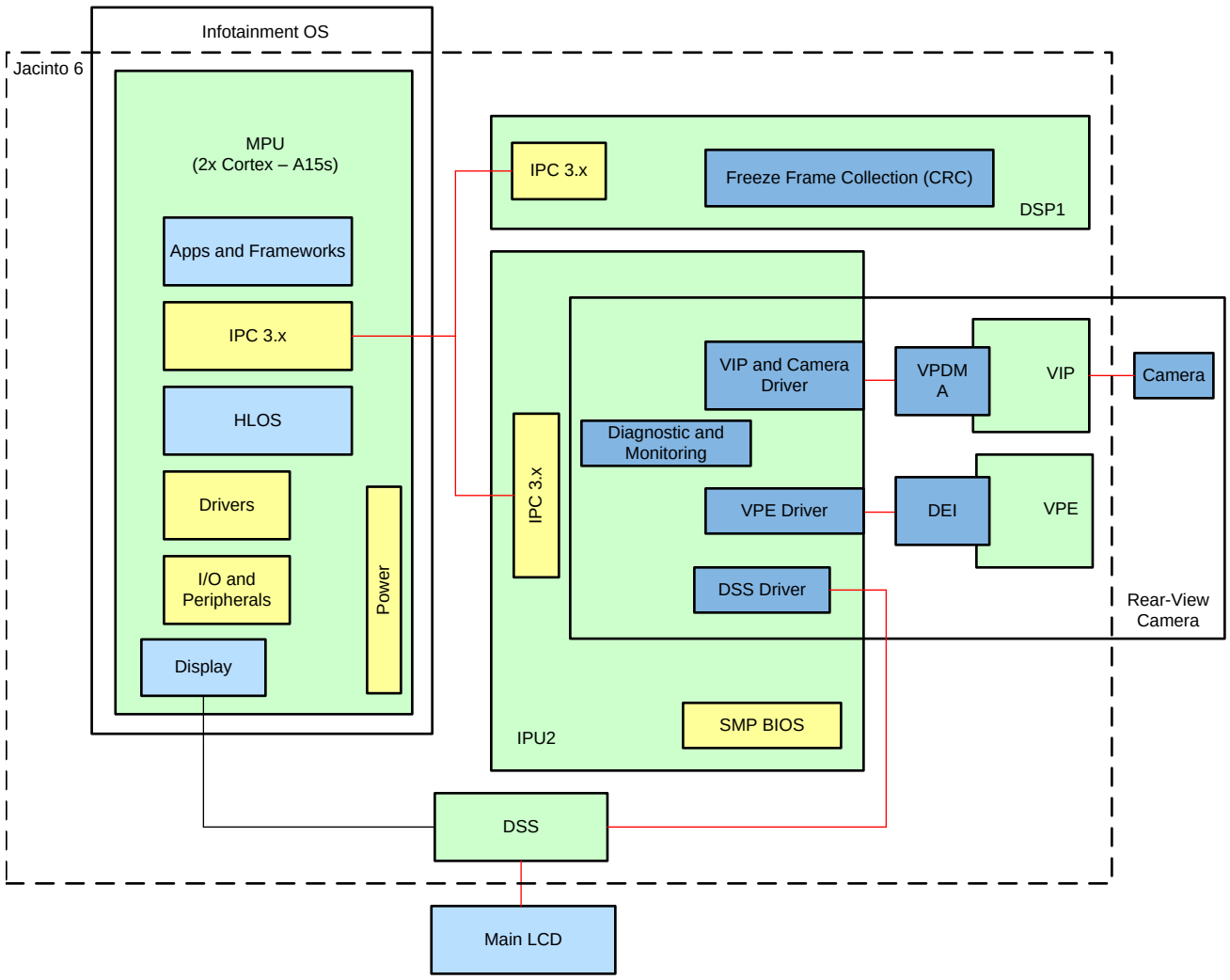

Figure 1. Robust RVC Architecture Diagram

The following are default Robust RVC integrated features:

- Initial diagnostics of the system configuration to ensure the system configuration is good and ready for Robust RVC
- Early Robust RVC to guarantee the camera get displayed within 2 seconds, even when Android HLOS fails to boot to the UI
- Guarantee RVC survival when the Android HLOS running on the A15/MPU crashes.

- Memory and peripherals firewall employed to protect camera and display paths for RVC
- Shared peripheral monitoring and correction, user warning
- DSP-based freeze-frame checks /w, a CRC calculation using a DSS WB of the RVC frame.
- Guarantee performance during high DDR load
- Guarantee performance during high interconnect load
- Glass-to-glass latency requirements of 100 milliseconds.
- Framework to support capture and display interrupt pacing tracking
- GPIO reverse-gear detection and conditional turn on and off of the A15 watchdog.

3 Robust RVC Software Components

The Robust RVC component involves modifications to the following software components at a block level (Figure 2).

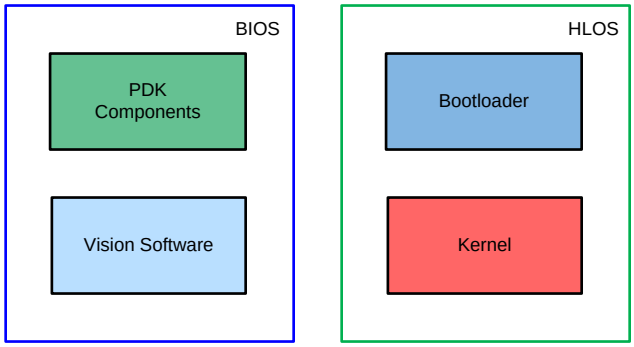

Figure 2. Robust RVC Software Components

- HLOS
  - Bootloader (U-boot): A single-stage boot that contains loading of the remote cores firmware images IPU2 and DSP1, initializing the display panel.
  - Kernel: Has the changes for skipping or adjusting the display initialization, and resolving the resource conflicts which are initialized by the vision components.
- BIOS
  - Vision_sdk: A framework to initiate the RVC use case. It forms a use-case chain with required links.
  - PDK components: PDK components comprise of Starterware drivers and BIOS BSP components. This component has all the base drivers required for the RVC use-case camera sensor, VIP, DSS, I2C and other drivers are enabled.

3.1 Workspace Locations

Table 2 lists the typical location for components in the Robust RVC.

$WORKSPACE is the location where the Robust RVC release package is installed.

Table 2. Robust RVC Workspace

Component	Location
U-boot	$WORKSPACE/u-boot
Kernel	$WORKSPACE/kernel-omap
Vision_sdk	$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk
PDK Components	$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/ti_components/drivers/pdk_01_xx_xx_xx

3.2 Robust RVC Use Case on Vision SDK

The Robust RVC, use-case specific files on top of the vision_sdk release package follow:

Use-case name: vip_single_rvc_cam_view_crc

```
$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/usecases/vip_single_rvc_cam_view_cr
c
```

Common chain where Robust RVC main use case is defined:

```
$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/common/chains_main_robust_rvc.c
```

3.3 Robust RVC Use Case Kernel Files

Robust RVC-specific configurations should go to the following dts files:

```
DRA75x/DRA74x: $WORKSPACE/kernel-omap/arch/arm/boot/dts/dra7-evm-robust-rvc.dts
DRA71x: $WORKSPACE/kernel-omap/arch/arm/boot/dts/dra71-evm-robust-rvc.dts
DRA72x: $WORKSPACE/kernel-omap/arch/arm/boot/dts/dra72-evm-robust-rvc.dts
DRA76x (with HDMI output): $WORKSPACE/kernel-omap/arch/arm/boot/dts/dra76-evm-robust-rvc.dts
DRA76x (with TFP410 adapter): $WORKSPACE/kernel-omap/arch/arm/boot/dts/dra76-evm-tfp410-robust-
rvc.dtb
```

3.4 Robust RVC Use Case U-Boot Files

Robust RVC specific U-boot configurations are defined in the following file:

```
$WORKSPACE/u-boot/board/ti/dra7xx/lateattach.c
$WORKSPACE/u-boot/board/ti/dra7xx/display.c
$WORKSPACE/u-boot include/configs/dra7xx_evm.h
```

4 Memory Mapping

Memory mapping of the required, various components of the Robust RVC follows. These entries are specific to the PROCESSOR_SDK_VISION_03_xx memory map.

4.1 Memory Mapping on Vision_SDK

The Robust RVC follows the Linux® memory maps for TDA2xx and DRA74x located at: $WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/build/tda2xx\mem_segment_definition_linux.xs.

Table 3 lists the DDR memory map at vision_sdk.

Table 3. Vision_sdk Memory Map

Component	Size	Address	Use
A15-Linux	64MB	0x80000000	
SR1 and NDK	256MB	0x84000000	SR1 + NDK
IPU2 – Bios	80MB	0x99000000	Vision-SDK FWK (12MB hole)
IPU1 – Bios	32MB	0x9e000000	IPU2 core
DSP1 – Bios	64MB	0xA1000000	RADIO (and Analytic algos)
DSP2 – Bios	32MB	0xA3000000	SRV Algos
SR2_BASE_ADDR		0xA9000000	
A15 – Linux		0xC0000000	(End of interleaving)

NOTE: Addresses and sizes mentioned in Table 3 are specific according to the PROCESSOR_SDK_VISION_03_xx release and are subject to change.

The required memories must be carved out in the kernel, as discussed in the next section.

4.2 Kernel Memory Carve-Out

Memories defined in vision_sdk memory map must be aligned and reserved in the kernel by doing carve-outs. The memories required by the Robust RVC use case follow:

- rvc_pool1 – The memory corresponding to SR1+NDK according to the vision_sdk memory segment map.
- rvc_pool2 – The memory corresponding to SR2 according to the vision_sdk memory segment map.
- ipu2_cma_pool – This memory corresponds to the IPU2-Bios segment in the vision_sdk memory map.
- dsp1_cma_pool – This memory corresponds to the DSP1-Bios segment in the vision_sdk memory map.

File location for DRA75x and DRA74x: $WORKSPACE/kernel-omap/arch/arm/boot/dts/dra7-evm-robust-rvc.dts

The following are reference entries in RVC-specific dts files, according to the PROCESSOR_SDK_VISION_03_xx memory map changes.

```
rvc_pool1: rvc1@0x84000000 {
        reg = <0x0 0x84000000 0x0 0x10000000>;
        status = "okay";
    };
    rvc_pool2: rvc2@0xA0000000 {
    reg = <0x0 0xA0000000 0x0 0x530000>;
    status = "okay";
    };
&ipu2_cma_pool {
    reg = <0x0 0x99000000 0x0 0x5000000>;
 };
&dsp1_cma_pool {
    reg = <0x0 0xA1000000 0x0 0x2000000>;
 };
```

4.3 U-Boot Memory Segment Definitions

The RVC uses IPU2 and DSP1 cores whose firmware is loaded from the bootloader with late-attach mode. Memory segments are defined in the bootloader. These segments should match the carve-outs defined in the kernel-dts and vision_sdk memory map.

Memory segments are defined in: $WORKSPACE/u-boot/board/ti/dra7xx/lateattach.c

```
#define DRA7_RPROC_CMA_BASE_IPU1          0x9e000000
#define DRA7_RPROC_CMA_BASE_IPU2          0x99000000
#define DRA7_RPROC_CMA_BASE_DSP1          0xa1000000
#define DRA7_RPROC_CMA_BASE_DSP2          0xa3000000
#define DRA7_RPROC_CMA_SIZE_IPU1          0x02000000
#define DRA7_RPROC_CMA_SIZE_IPU2          0x05000000
#define DRA7_RPROC_CMA_SIZE_DSP1          0x02000000
#define DRA7_RPROC_CMA_SIZE_DSP2          0x02000000
```

5 Configuring Robust RVC With IPU1 and DSP1

By default, the Robust RVC is configured to use IPU2 and DSP1. The following subsections detail the considered changes needed, in addition to PROCESSOR_SDK_VISION_03_xx, to configure the Robust RVC on IPU1 and DSP1.

5.1 Vision_sdk

5.1.1 Configuring IPU_PRIMARY_CORE

To configure IPU_PRIMARY_CORE, modify on: $WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/configs/tda2xx_evm_robust_rvc/cfg.mk.

Make the following changes:

From:

```
PROC_IPU1_0_INCLUDE=no
PROC_IPU1_1_INCLUDE=no
PROC_IPU2_INCLUDE=yes
PROC_A15_0_INCLUDE=no
PROC_DSP1_INCLUDE=yes
```

To:

```
PROC_IPU1_0_INCLUDE=yes
PROC_IPU1_1_INCLUDE=no
PROC_IPU2_INCLUDE=no
PROC_A15_0_INCLUDE=no
PROC_DSP1_INCLUDE=yes
```

Select the IPU primary core from the available IPU1 and IPU2 sub-system, make the following changes:

From:

```
IPU_PRIMARY_CORE=ipu2
IPU_SECONDARY_CORE=ipu1_0
```

To:

```
IPU_PRIMARY_CORE=ipu1_0
IPU_SECONDARY_CORE=ipu2
```

To disable IPUMM, make the following change:

From:

```
IPUMM_INCLUDE=yes
```

To:

```
IPUMM_INCLUDE=no
```

NOTE: If IPUMM needs to be part of the IPU1 image, then IPUMM_INCLUDE must be set to yes. Along with it, the IPU1 memory configuration must be aligned to accommodate the IPUMM memory configuration, for example, the carve out (see Section 4).

5.1.2 Configuring Remote Core

Modify on:
$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/usecases/vip_single_rvc_cam_view_crc/cfg.mk.

To configure the remote core to be included, make the following changes:

From:

```
NEED_PROC_IPU2=yes
```

To:

```
NEED_PROC_IPU2=no
NEED_PROC_IPU1_0=yes
```

5.2 Kernel-Modifying dts Configurations

Due to the remote core change from IPU2 to IPU1, the Robust RVC, kernel-specific, DTS configuration must be updated as well.

Reference changes to be done in the Robust RVC dts file for changing the core from IPU2 to IPU1 follow.

Modify for:

- DRA75x and DRA74x: dra7-evm-robust-rvc.dts
- DRA71x: dra71-evm-robust-rvc.dts
- DRA72x: dra72-evm-robust-rvc.dts
- DRA76x: dra76-evm-robust-rvc.dts or dra76-evm-tfp41-0-robust-rvc.dts (depending on display output)

From:

```
&mbox_ipu2_ipc3x {
    ti,no-reset-on-init;
     ti,no-idle-on-init;
 };
&mmu_ipu2 {
{
     ti,late-attach;
     ti,no-reset-on-init;
     ti,no-idle-on-init;
 };
&ipu2 {
    ti,late-attach;
     ti,no-reset-on-init;
     ti,no-idle-on-init;
```

To:

```
&mbox_ipu1_ipc3x {
     ti,no-reset-on-init;
     ti,no-idle-on-init;
 };
&mmu_ipu1 {
{
     ti,late-attach;
     ti,no-reset-on-init;
     ti,no-idle-on-init;
 };
&ipu1 {
    ti,late-attach;
     ti,no-reset-on-init;
     ti,no-idle-on-init;
};
```

5.3 U-Boot Modifications

5.3.1 Modifying eMMC Partitions

IPU and DSP firmware images are flashed to the embedded MultiMedia card (eMMC) partition. The partition sizes are different than the regular kernel baseline. Partition sizes can be optimized.

To modify the partition sizes:

1. Update the following file:

   $WORKSPACE/u-boot/include/configs/dra7xx_evm.h
2. Modify the following lines with the desired sizes:

   ```
   "name=ipu1,size=7M,uuid=${uuid_gpt_ipu1};" \ "name=dsp1,size=1M,uuid=${uuid_gpt_dsp1};" \
   ```
3. Rebuild the bootloader and reflash the images according to the flashing instructions in the release document.
4. Reformat the eMMC with the new sizes and re-generating the userdata image to fit the userdata partition. Follow the release document guide reformatting the eMMC.

5.3.2 Modifying Booting Cores

Default cores to be loaded from U-boot must be updated with IPU1.

Modify: $WORKSPACE/U-boot/common/spl/spl.c

From:

```
cores_to_boot[] = { IPU2, DSP1 };
```

To:

```
cores_to_boot[] = { IPU1, DSP1 };
```

6 Robust RVC Diagnostics

The Robust RVC has a feature that allows it to perform system diagnostics by monitoring the various module registers it operates on. The Robust RVC performs check on more registers on VIP and DSS. In addition, the Robust RVC is not only checking the register, but it also performs an automatic correction on those registers which have a static configuration value.

The Robust RVC diagnostics software configuration is at:

$WORKSPACE/ PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/alg_plugins/rvcDiags/.

6.1 Automatic Register Correction

To turn on or off the automatic register correction, modify the usecase file:

$WORKSPACE/
PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/usecases/vip_single_rvc_cam_view_crc/
chains_vipSingleRvcCamCrc_Display.c

To enable Robust RVC Diagnostics automatic correction:

```
pUcObj->Alg_RvcDiagnosticPrm.autoCorrectFlag = TRUE
```

To disable Robust RVC Diagnostics automatic correction:

```
pUcObj->Alg_RvcDiagnosticPrm.autoCorrectFlag = FALSE
```

6.2 Robust RVC Diagnostics Reference Data

The reference values used for register checking and completing corrections are specified at:

$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/alg_plugins/rvcDiags/rvcDiagnostic_algLink_priv.h

Table 4 lists the reference values for DSS configurations to support LCD1 for DRA75x and DRA74x, DRA72x, DRA71x, and DRA76x (with HDMI output), with an RVC display size of 720 × 480.

Table 4. Register Diagnostics

Register Name	DRA75x and DRA74x	DRA71x	DRA72x	DRA76x With HDMI
VIP_MAIN	0x2	0x2	0x0	0x2
VPDMA_SETUP	0x0	0x0	0x0	0x0
DSS_DISPC_TIMING_H1	0x01F06F0F	N/A	0x01F06F0F	N/A
DSS_DISPC_TIMING_H3	N/A	0x02F02F1F	N/A	N/A
DSS_DISPC_POL_FREQ1	0x0	0x0	0x0	0x0
DSS_DISPC_GLOBAL_ALPHA	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFFFFFF
DSS_DISPC_SIZE_LCD1	0x04AF077F	N/A	0x04AF077F	N/A
DSS_DISPC_SIZE_LCD3	N/A	0x031F04FF	N/A	N/A
DSS_DISPC_VID2_POSITION	0x00640258	0x00640118	0x00640258	0x00640258
DSS_DISPC_VID2_SIZE	0x01DF02CF	0x01DF02CF	0x01DF02CF	0x01DF02CF
DSS_DISPC_VID2_ATTRIBUTES	0x0608880B	0x8608880B	0x0608880B	0x0609880B
DSS_DISPC_VID2_BUF_THRESHOLD	0x07FF07F8	0x07FF07F8	0x07FF07F8	0x07FF07F8
DSS_DISPC_VID2_ROW_INC	0x1	0x1	0x1	0x1
DSS_DISPC_VID2_FIR	0x04000400	0x04000400	0x04000400	0x04000400
DSS_DISPC_VID2_PICTURE_SIZE	0x01DF02CF	0x1DF02CF	0x01DF02CF	0x01DF02CF
DSS_DISPC_VID3_POSITION	0x00000258	0x00000118	0x00000258	0x00000258
DSS_DISPC_VID3_SIZE	0x01DF02CF	0x01DF02CF	0x01DF02CF	0x01DF02CF
DSS_DISPC_VID3_ATTRIBUTES	0x0A288A35	0x8A288A35	0x0A288A35	0x0A298A35
DSS_DISPC_VID3_ATTRIBUTES2	0x0	0x0	0x0	0x0
DSS_DISPC_VID3_ROW_INC	0x1	0x1	0x1	0x1
DSS_DISPC_VID3_PIXEL_INC	0x1	0x1	0x1	0x1
DSS_DISPC_VID3_PICTURE_SIZE	0x01DF02CF	0x01DF02CF	0x01DF02CF	0x01DF02CF
CTRL_CORE_IPU2_IRQ_49_50	0x15B	N/A	N/A	N/A

NOTE: Diagnostic auto-correction of the Robust RVC must be turned off when debugging with the debug interface. If parameter configuration changes are performed, the RVC diagnostic module reference values must be updated appropriately.

The next section describes how to modify the reference values when display sizes are changed.

7 Configuration of Robust RVC Display Frame Size

To configure Robust RVC frames to a display size for an LCD 10'' OSD panel on the DRA7xx, the following shows an example for 1920" × 1200".

7.1 Configuring VID2 and VID3 Start Positions

To configure the start positions, modify the following:

$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/usecases/vip_single_rvc_cam_view_crc/chains_vipSingleRvcCamCrc_Display.c

Default configurations for the VID2 and VID3 start position X and Y, display width and height:

```
#define VID2_POSX        (280)
#define VID2_POSY        (45)
#define VID3_POSX        (280)
#define VID3_POSY        (10)
#define CAPTURE_DISPLAY_WIDTH      (720)
#define CAPTURE_DISPLAY_HEIGHT     (480)
```

Change start positions to:

```
#define VID2_POSX        (0)
#define VID2_POSY        (0)
#define VID3_POSX        (0)
#define VID3_POSY        (0)
#define CAPTURE_DISPLAY_WIDTH      (1280)
#define CAPTURE_DISPLAY_HEIGHT     (800)
```

7.2 Configuring the Robust RVC Diagnostics

To configure the Robust RVC diagnostic information, modify the following:

$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/alg_plugins/rvcDiags/rvcDiagnostic_algLink_priv.h,

Default register reference configurations for the DSS module configured are at:

```
// DISPC Module
    // 8: DSS_DISPC_VID2_POSITION (POSX = 280; POSY = 45)
    {SOC_DISPC_BASE, DSS_DISPC_VID2_POSITION, 0xFFFFFFFF, 0, 0x002D0118},
    // 9: DSS_DISPC_VID2_SIZE (SIZEY = 479; SIZEX = 719; 720x480 VID2 size)
    {SOC_DISPC_BASE, DSS_DISPC_VID2_SIZE, 0xFFFFFFFF, 0, 0x01DF02CF},
    // 10: DSS_DISPC_VID2_ATTRIBUTES
    {SOC_DISPC_BASE, DSS_DISPC_VID2_ATTRIBUTES, 0xFFFFFFFF, 0, 0x0620086B},
    {SOC_DISPC_BASE, DSS_DISPC_VID3_POSITION, 0xFFFFFFFF, 0, 0x000A0118},
    // 16: DSS_DISPC_VID3_SIZE
    {SOC_DISPC_BASE, DSS_DISPC_VID3_SIZE, 0x0FFF0FFF, 0, 0x01DF02CF},
     // 17: DSS_DISPC_VID3_ATTRIBUTES
    {SOC_DISPC_BASE, DSS_DISPC_VID3_ATTRIBUTES, 0xFFFFFFFF, 0, 0x0A200A35},
```

Modify the DSS register reference configurations to the following:

```
// DISPC Module
        // 8: DSS_DISPC_VID2_POSITION (POSX = 0; POSY = 0)
        {SOC_DISPC_BASE, DSS_DISPC_VID2_POSITION, 0xFFFFFFFF, 0, 0x00000000},
        // 9: DSS_DISPC_VID2_SIZE (SIZEY = 799; SIZEX = 1279; 1280x480 VID2 size)
        {SOC_DISPC_BASE, DSS_DISPC_VID2_SIZE, 0xFFFFFFFF, 0, 0x031F04FF},
        // 10: DSS_DISPC_VID2_ATTRIBUTES
        {SOC_DISPC_BASE, DSS_DISPC_VID2_ATTRIBUTES, 0xFFFFFFFF, 0, 0x0608880B},
        // 15: DSS_DISPC_VID3_POSITION
        {SOC_DISPC_BASE, DSS_DISPC_VID3_POSITION, 0xFFFFFFFF, 0, 0x00000000},
        // 16: DSS_DISPC_VID3_SIZE (SIZEY = 799; SIZEX = 1279; 1280x800 VID3 size)
        {SOC_DISPC_BASE, DSS_DISPC_VID3_SIZE, 0x0FFF0FFF, 0, 0x031F04FF},
        // 17: DSS_DISPC_VID3_ATTRIBUTES
        {SOC_DISPC_BASE, DSS_DISPC_VID3_ATTRIBUTES, 0xFFFFFFFF, 0, 0x0A288A35},
```

8 Support for New Camera/Sensor

To add support for a new sensor, follow the instructions in the PDK Driver Porting Guide.

8.1 Support for OV1635 Sensor

The OV10635 sensor is a driver supported in the PDK components, and vision_sdk framework has functionality for configuring this sensor and its dependencies. This section provides the details on how to add support for the OV10635 camera which is to RVC usecase. RVC usecase supports TVP5158 sensor by default.

Use the following procedure to configure the OV10635 sensor.

8.1.1 RVC Use-Case Chain for Analog-to-Digital Sensor

Modify the RVC use-case creation file to remove the VPE De-interlacer:

$WORKSPACE/PROCESSOR_SDK_VISION_03_xx//vision_sdk/apps/src/rtos/usecases/vip_single_rvc_cam_view_crc/chains_vipSingleRvcCamCrc_Display.txt

Make the following change:

From:

```
UseCase: chains_vipSingleRvcCamCrc_Display

Capture ->  Alg_RvcDiagnostic -> VPE_dei -> Display_Video

Capture_dsswb -> Alg_SwCrc (DSP1)

GrpxSrc -> Display_Grpx
```

To:

```
UseCase: chains_vipSingleRvcCamCrc_Display

Capture ->  Alg_RvcDiagnostic -> Display_Video

Capture_dsswb -> Alg_SwCrc (DSP1)

GrpxSrc -> Display_Grpx
```

The following files are generated:

```
chains_vipSingleRvcCamCrc_Display_priv.c

chains_vipSingleRvcCamCrc_Display_priv.h

chains_vipSingleRvcCamCrc_Display_img.txt

chains_vipSingleRvcCamCrc_Display.jpg
```

8.1.2 Sensor Selection

The OV10635 sensor is selected by modifying:

$WORKSPACE/PROCESSOR_SDK_VISION_03_xx/vision_sdk/apps/src/rtos/common/chains_main_robust_rvc.c,

```
gChains_usecaseCfg.captureSrc = CHAINS_CAPTURE_SRC_OV10635;
```

Modify the RVC usecase file: chains_vipSingleRvcCamCrc_Display.c

- To remove VPE_DEI configurations
- To adjust input height and width of the capture sensor

```
#define CAPTURE_SENSOR_INPUT_WIDTH      (1280)
#define CAPTURE_SENSOR_INPUT_HEIGHT     (720)
```

NOTE: RVC diagnostics reference values must be changed according to the updated VIP and DSS configurations.

9 Freeze Frame CRC Check

The Robust RVC provides a feature to identify a freeze frame, based on the CRC checks on the previous frame and current frame. The CRC check is performed on DSS WB data from the overlay output. This section provides the details for CRC configuration using DSS WB mode.

The CRC check is performed on the region of interest where the Robust RVC must be displayed.

9.1 Capture Mode

CaptureMode defines the frame rate of the DSS WB pipe. To change CaptureMode, modify the Robust RVC use-case file: chains_vipSingleRvcCamCrc_Display.c

```
pPrm->dssWbInst[0].dssWbOutputPrms.wbCaptureMode = 3;
```

The previous configuration defines one out of every three frames captured by the DSS WB.

10 Firewall Configuration

A memory and peripherals firewall is employed to protect the camera and display paths for the Robust RVC. The firewall configuration is supported only for HS devices. Implementation of the firewall feature is done using PPA HAL APIs available in the secdev-dra7xx_std packages.

Enable firewall involves modifications to:

- U-boot
- secdev-dra7xx_std_xx package

This section covers the modifications to be done at U-boot to configure the L3 and L4 firewall.

10.1 L3 Firewall Configuration

L3 firewall configurations are done to protect the memory, firmware, and resources used by the Robust RVC. The resource table, IPC memory, and trace buffer areas are not protected by the MPU firewall.

10.1.1 MreqDomain

Firewall inband attributes have MreqDomain, which can be used with HS silicon. This isolates the initiators which share the same connection ID.

MreqDomain indicates the domain in which the access was generated. It enables the set up of an isolated subsystem of initiator/target that can be protected from the rest of the platform.

Table 5 lists MReqDomain configured with the Robust RVC release changes for the secdev-dra7xx_std_xx package.

Table 5. Domain ID

Domain ID	Initiators
DOMAIN_1	IPU2 and IVA
DOMAIN_2	DSP1
DOMAIN_3	VIP and VPE
DOMAIN_4	DSS
DOMAIN_0	All other initiators

10.1.2 EMIF Firewall Configuration

The EMIF firewall is setup for the regions with access permissions for required initiators, as shown in Figure 3.

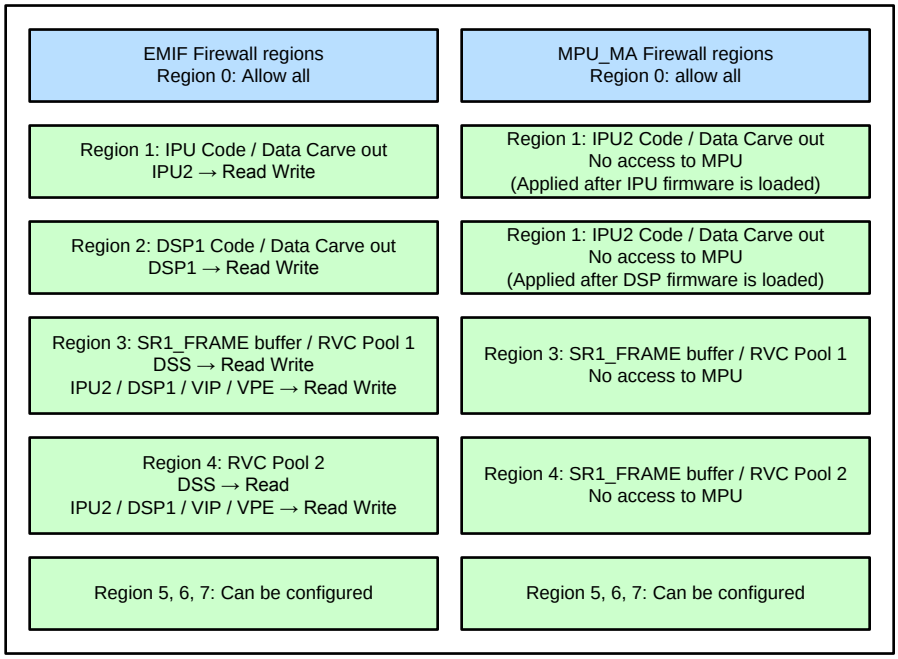

Figure 3. Robust RVC EMIF and MPU Firewall Regions

Access permissions for the EMIF firewall are done in:

$WORKSPACE/Uboot/arch/arm/cpu/armv7/omap-common/emif-common.c M emif_fw_config()

To set the permission for Region 1:

```
        /* Region 1: (IPU2 Code/Data reserved memory) Allow access to IPU2/IVA  only
        * Address range for IPU2 CMA Pool 0x99000000 for size 0x05000000
        */
        accessPerm    = L3_ALL_PERMISSIONS
                                    | L3_DOMAIN1_FUNCTIONAL
                                    | L3_DOMAIN1_DEBUG;
        initiatorPerm = FW_INITIATOR_IVA1
                                    |FW_INITIATOR_DMA_IPU;

        if (0 != secure_emif_firewall_setup(1, 0x1, 0x99000000, 0x05000000,

accessPerm, initiatorPerm))
```

10.1.3 MPU Firewall

The MPU firewall is setup for the regions with access permissions for required initiators, as shown in Figure 3.

Access permissions for MPU firewall are done in:

$WORKSPACE/Uboot/ common/spl/spl.cM enable_firewall_non_emif ()

To set the permission for region 3 for the MPU firewall:

```
//Set the MPU firewall for RVC frame buffers RVC pool1
if(0 != secure_mpu_firewall_lock(3, FW_INITIATOR_MPU, FW_INITIATOR_READ,

0x84000000,0x10000000))
```

NOTE: Figure 3 shows the regions which can be additionally configured as Region 1 for IPU2 CMA Pool. As part of Robust RVC release only Region 3 is configured for MPU firewall access.

10.2 L4 Firewall Configuration

The L4 firewall is setup to protect the VPE and VIP CFG space. Figure 4 shows the regions configured for the L4 Per 3 firewall.

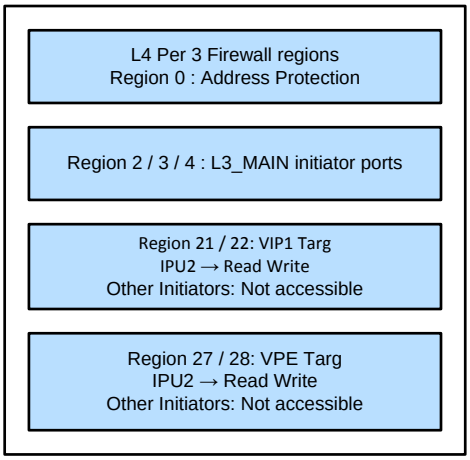

Figure 4. Robust RVC L4 Firewall Regions

The L4 firewall can be enabled by calling $WORKSPACE/Uboot/ common/spl/spl.c -> enable_firewall_non_emif () -> secure_l4per3_firewall_lock().

The L4 firewall region is configured with the Robust RVC release changes for the secdev-dra7xx_std_xx package.

11 Display Panel Adaptation

For the Robust RVC to meet the display requirement of < 2 seconds, the display panel is started at u-boot instead of the M4 firmware.

At u-boot, common/spl/spl.c, board_init_r() calls the internal function spl_load_image(), where the setup of the display panel is configured through board/ti/dra7xx/display.c::spl_setup_display().

Because the display panel is started and configured at u-boot, the display panel timing must be synchronized between the u-boot, vision-sdk, and Android kernel. The configuration location for the OSD display panel follows.

- At the u-boot, see: board/ti/dra7xx/display.c and tlc_board_data, for example: tlc_osd_2045_10_inch_data
- At the Vision SDK, see: vision_sdk/apps/src/rtos/usecases/common/chains_common.c::ChainsCommon_SetDctrlConfig() and ti_components/drivers/pdk_xx_xx_xx_xx/packages/ti/drv/pm/src/pmlib/prcm/V0/pmlib_videopll_data.c on gVidPllPreCalcTbl[], to ensure the PLL settings match the clock frequency that is being set.
- At the kernel, see: arch/arm/boot/dts/dra7x-evm-lcd-osd.dtsi on the panel-timing configuration.

12 Display Sharing

The Display Subsystem (DSS) is a common resource between the MPU and IPU subsystems, and DSS is the central component to managing display-screen content. On the DRA7x, the DSS hardware block has a total of 4 pipelines (1 GFX and 3 VID pipes), to simultaneously process 4 buffers. Per design, the RVC on the IPU software takes priority for the DSS resources. The RVC uses two pipes (VID2 and VID3), one for the rear-view camera, and the other pipe is currently used for notification, for example, Frame-Freeze detection, but the user can modify the pipe to display the steering guide, see Figure 5.

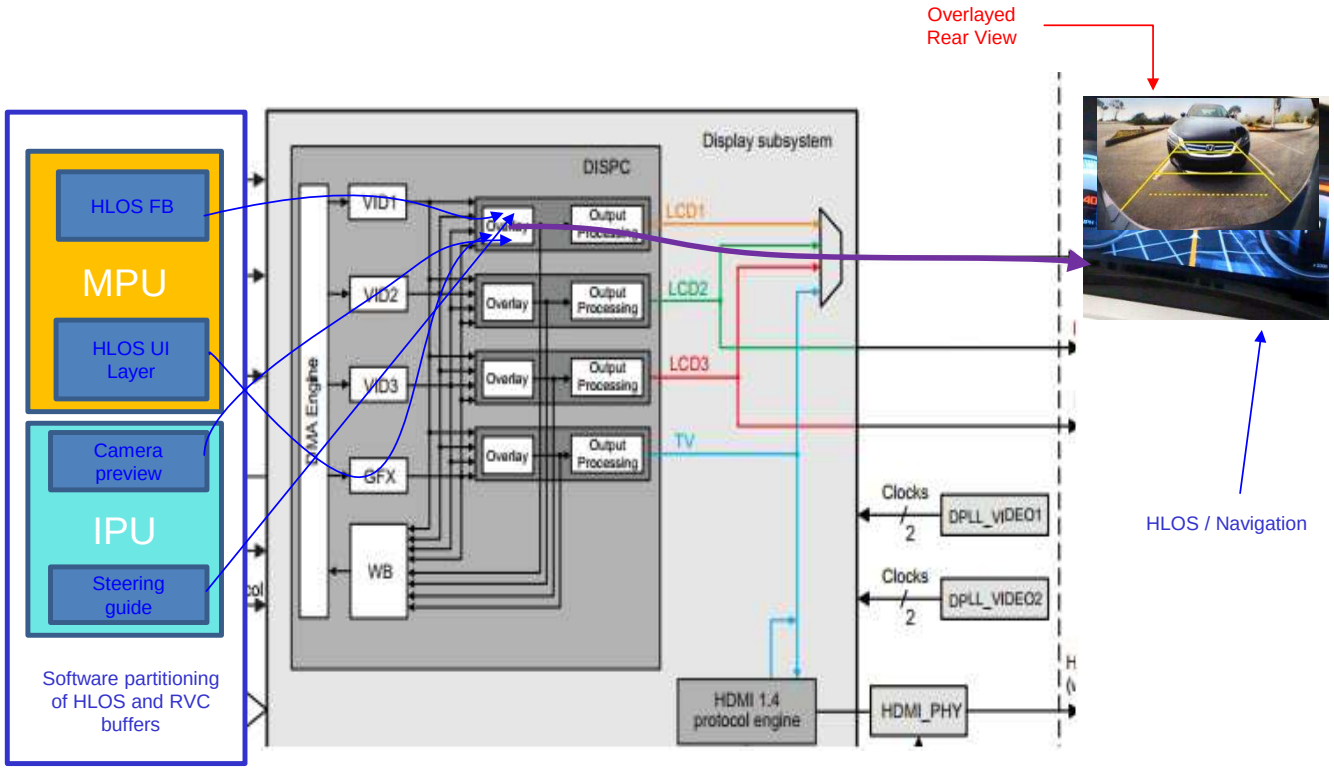

Figure 5. Display Sharing Block Diagram

The Android HLOS display manager skips the DSS pipes reserved for the RVC. This feature is done in the kernel, by configuring the number of DSS overlays to the 2 pipes only (ti_config_fragments/audio_display.cfg). Additionally, the Android HLOS DSS driver is also modified to skip reset and clock-configuration sequences, to co-exist with the IPU-based DSS driver.

The change on the Android kernel is in:

```
drivers/gpu/drm/omapdrm/dss/dss.c
drivers/gpu/drm/omapdrm/dss/core.c
drivers/gpu/drm/omapdrm/dss/dispc.c
drivers/gpu/drm/omapdrm/dss/dpi.c
drivers/gpu/drm/omapdrm/omap_irq.c
include/video/omapdss.h
```

On the Vision SDK M4 side, the change is in:

```
drivers/pdk_01_08_01_06/packages/ti/drv/vps/src/vpslib/dispcore/src/vpscore_dctrl.c
drivers/pdk_01_08_01_06/packages/ti/drv/vps/src/vpslib/hal/src/vpshal_dssDispcOvly.c
drivers/pdk_01_08_01_06/packages/ti/drv/vps/src/vpslib/hal/vpshal_dssDispcOvly.h
```

13 FAQs

- *How do I check traces for remote cores which run vision_sdk components?*

  Robust RVC uses IPU2 and DSP1 cores. Remote core traces can be checked by debugfs. Run the following commands:

  ```
  IPU2: $cat /d/remoteproc/remoteproc1/trace0
  DSP1: $cat /d/remoteproc/remoteproc2/trace0
  ```

- *How do I add traces to vision_sdk components for debugging?*

  To add the traces in vision_sdk, use Vps_printf().

- *How do I print statistics using vision_sdk?*

  To get the load statistics and L3 BW stats on vision_sdk, define the following macro in the use-case file: chains_vipSingleRvcCamCrc_Display.c

  ```
  #define PRINT_STATISTICS
  ```

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