VAYU-BSP MigrationGuide

Introduction

This document is the Migration Guide for BSP.

The purpose of this document is to:

- Explain differences between HDVPSS (TI81xx) and BSP (Tda2xx) packages
- Describe some of the architectural and functional improvements in BSP over HDVPSS and gives reasons why the changes are made.
- Give information on how application writers can migrate

Before reading this document, it is strongly advised to first go through the BSP User Guide

Terms and Abbreviations

Abbreviation	Description
BSP	BIOS Support Package - BIOS Drivers
CCS	Code Composer Studio
CGTools	Code Gen Tools, e.g. Compiler, Linker, Archiver
DSS	Display Sub-System
HDVPSS	High Definition Video Processing Sub-System - HDVPSS Drivers
ISS	Imaging Sub-System
VIP	Video Input Port
VPE	Video Processing Engine
VPS	Video Processing Sub-System containing all video related peripherals

Migration from HDVPSS

HDVPSS is the BIOS drivers for the HDVPSS module which encompasses VIP, display and memory to memory paths lie NSF, DEI and SC. Supported devices include TI816x, TI814x, DM385, J5ECO and TI8149 (Cent-EVE)

BSP (BIOS Support Package) is BIOS drivers for the various peripherals present in next generation of SOCs namely TDA2xx devices. This package covers the following

- VIP capture driver covering the 3 independent VIP blocks
- VPE M2M driver for the VPE block
- DSS display driver for the display sub system
- Serial drivers: McASP, McSPI, I2C and UART

The below sections has the detailed migration information from HDVPSS to BSP.

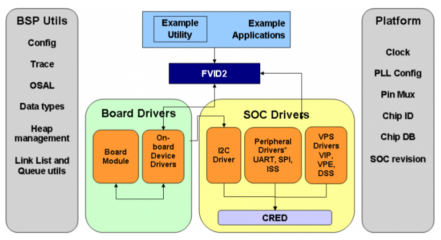
Reason for API and Package Changes

Below are the main reasons for the changes made from HDVPSS and BSP packages

- Significant changes in hardware
  - No longer HDVPSS
  - VIP and VPE in TDA2xx is different
  - VIP and VPE are independent from each other unlike HDVPSS where all video sub-modules has common VPDMA and CLKC controls
  - DSS (from OMAP5) is different from HDVPSS display
- Re-design of video driver (VPS) into FVID2/OS dependent driver layer and hardware/OS independent core layers
  - Less effort to implement drivers for newer devices
  - Consistent interface for any new device
  - Enables OS-independent layers for sharing with DV, IP validation and debugging
  - Easier to reproduce customer use cases with core layer
  - Reuse of core and HAL layers for device starterware
  - Possible reuse of 50% of code across other OS like QNX
- To make software simpler and match IP structuring
- Reworked the directory structure to make it easier to
  - Support custom boards
  - Support new on-board video peripherals
- Incorporated feedback received from customers and application developers

Changes in Package, Directory Structure and Overall Design

Modules Block Diagram



BSP Module Block Diagram

Above is a pictorial representation of various modules present in the BSP.

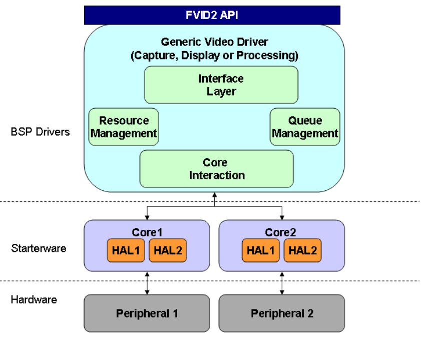
- Common/Utils:
  - Contains utils functions, trace modules etc...
  - This module is used across all other modules
- Platform:
  - This module contains all the SOC related information like chip revision, PRCM, PLL config, pinmux etc..
- FVID2:
  - Device and platform independent video driver layer
- SOC Drivers:
  - Contains all the SOC level peripheral drivers
  - VPS: VIP, DSS and VPE
  - Serial: UART, I2C, McASP, McSPI

    - These drivers are board independent and no information related to a particular board/EVM is assumed inside these drivers.
- Board/Device Drivers:
    - Board specific drivers and board specific information are abstracted here
    - Drivers for all board peripherals like AIC31, TVP5158 and sensors are present here

The high level changes compared to HDVPSS are

- FVID2: In HDVPSS, FVID2 was part of VPS driver. Doing a FVID2 init did the VPS init as well. Now this is moved out of VPS module and made device and platform independent. Hence application has to call Fvid2_init() and Vps_init() separately.
- Platform: In HDVPSS, platform contains all the board related and SOC related functions. Now this is split into two modules: *platform* which does only SOC level configuration and *boards* which has the board/EVM specific configuration. The inter dependency is removed. Hence porting of the BSP package to a customer board is made easier.

Video Driver Stack

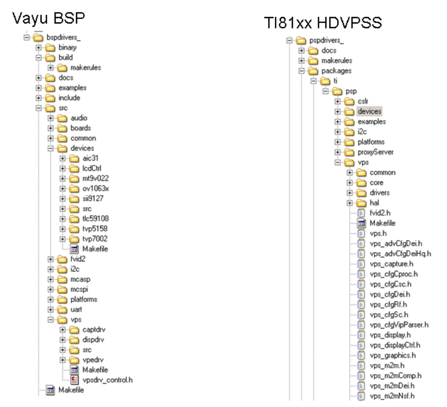


Video Driver Stack

Above is a pictorial representation of current design of video driver stack in BSP. There is a significant change in the internal design of the video drivers namely VIP, DSS and VPE.

- Video drivers (VPS) are re-designed into FVID2/OS dependent driver layer (present in BSP) and hardware/OS independent core layers or LLD (present in starterware)
- The FVID2 driver layer of the video peripherals takes of the queuing and resource management apart from managing the LLDs present in the starterware package.
- The hardware specific core/LLD present in the starterware package takes care of all device specific configuration and programming.
- The interaction between the driver and core layer is through a standard set of APIs allowing the same driver to interact with different set of peripheral cores like VIP and ISS core having same FVID2 capture driver layer.

Directory Structure Overview


Vayu BSP

```
bspdrivers_
    binary
    build
        makerules
    docs
    examples
    include
    src
        audio
        boards
        common
        devices
            aic31
            lcdCtrl
            mt9v022
            ov1063x
            sii9127
            src
            tlc59108
            tvp5158
            tvp7002
            Makefile
        fvid2
        i2c
        mcasp
        mcspi
        platforms
        uart
        vps
            captdrv
            dispdrv
            src
            vpedrv
            Makefile
            vpsdrv_control.h
    Makefile
```

TI81xx HDVPSS

```
pspdrivers_
    docs
    makerules
    packages
        ti
            psp
                cslr
                devices
                examples
                i2c
                platforms
                proxyServer
                vps
                    common
                    core
                    drivers
                    hal
                    fvid2.h
                    Makefile
                    vps.h
                    vps_advCfgDei.h
                    vps_advCfgDeiHq.h
                    vps_capture.h
                    vps_cfgCproc.h
                    vps_cfgCsc.h
                    vps_cfgDei.h
                    vps_cfgRf.h
                    vps_cfgSc.h
                    vps_cfgVipParser.h
                    vps_display.h
                    vps_displayCtrl.h
                    vps_graphics.h
                    vps_m2m.h
                    vps_m2mComp.h
                    vps_m2mDei.h
                    vps_m2mNsf.h
```

Directory Structure: BSP vs HDVPSS

- **bsp\binary**:
  - All the build files and executables are generated here
- **bsp\build**:
  - Contains all build related make files
- **bsp\docs**:
  - Contains user guide, API guide, release notes, migration guide etc…
  - Contains Gel files for various platforms
  - Contains test image files
- **bsp\examples**:
  - Application level utility functions which could be used by customers as is or with minimal modification
  - Sample applications for various drivers
- **bsp\include**:
  - Contains the application interface header files for all the modules
- **bsp\src\audio**:
  - Generic audio driver to stream data to/fro from application to actual audio transport stream drivers like McASP
- **bsp\src\board**:
  - List of on-board devices
  - Board auto-detect
  - Special board level programming
- **bsp\src\common**:
  - Trace functions
  - OS abstraction layer, Data types
  - Heap management like BSS memory and descriptor memory
  - Common utility functions like link list, queue etc…
- **bsp\src\devices**:
  - All on-board device drivers like video encoders, video decoders and sensors
- **bsp\src\fvid2**:
  - FVID2 driver API and driver level interface
- **bsp\src\i2c**:
  - I2C driver files
- **bsp\src\mcasp**:

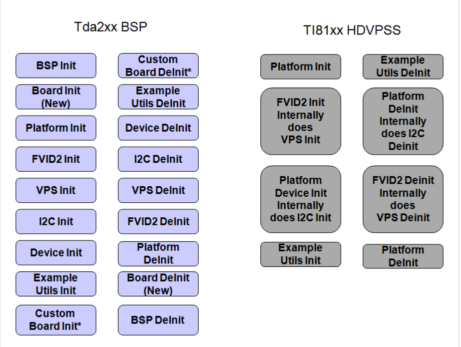
  - McASP driver files
- **bsp\src\mcspi**:
  - McSPI driver files
- **bsp\src\platforms**:
  - Top level directory for all supported platforms
- **bsp\src\uart**:
  - UART driver files
- **bsp\src\vps**:
  - Video driver package containing VIP, VPE and DSS drivers including FVID2 drivers, core drivers and HAL layer

The high level changes compared to HDVPSS are

- All the include header files were at the each module top level path in HDVPSS. In BSP, all the application header files are moved to *<root>/include* under each module directory separately.
- The actual driver source files are moved to *<root>/src* folder compared to HDVPSS where they were present at *<root>/packages/ti*
- Examples are moved to top level out of the driver source folder to make them independent. In HDVPSS, they were part of the driver source folder at *<root>/packages/ti/examples*
- Build make files are moved to *<root>/build/makerules*
- The generated obj, lib and executable files are part of *<root>/binary* folder. In HDVPSS they were under *<root>/build* folder

Detailed Changes

Init/deinit Sequence



Init/deinit: BSP vs HDVPSS

The above image shows the difference in the init and deinit sequence between the BSP and HDVPSS stacks.

- All modules are made independent from each other
  - FVID2 and VPS are made independent
  - Platform and device are made independent
  - Device and I2C are made independent. Now I2C is a standalone BIOS IOM driver module
- BSP modules stack is shown in the BSP block diagram in earlier section
- Interdependency of modules is removed. Now dependency is structured (top to bottom)
- All module init and deinit should be called separately

FVID2 Module

FVID2 Module Changes

Changes	TI81xx HDVPSS	Tda2xx BSP
Init/deinit	Internally calls VPS init	Independent module VPS init/deinit should be called separately
Interface File	Code Composer Studio	
Interface File	psp/vps/fvid2.h	bsp/include/fvid2/fvid2.h
Function and structure prefix	FVID2_	Fvid2_
Addition	NA	Interface Mode - Fvid2_VideoIfMode Interface Width - Fvid2_VideoIfWidth Init functions for all structures to support backward compatibility
Replacement	NA	channelNum to chNum Crop config - Fvid2_CropConfig (from Vps_CropConfig) Position Config - Fvid2_PosConfig (from Vps_PosConfig)

VPS Module

VPS Module Changes

Changes	TI81xx HDVPSS	Tda2xx BSP
Init/deinit	FVID2 init Internally calls VPS init	Independent module VPS init/deinit should be called separately
Init Parameter	NULL	Parameters to enable address translation of descriptor memory (virtToPhys) Parameters to enable cache operation if descriptors are present in cacheable section
Interface Files	psp/vps/*.h Capture – vps_capture.h M2M DEI – vps_m2m.h, vps_m2mDei.h Display – vps_display.h, vps_graphics.h, vps_displayCtrl.h	bsp/include/vsp/*.h Capture – vps_capture.h, vps_captureVip.h M2M VPE – vps_m2m.h, vps_m2mVpe.h Display – vps_display.h, vps_displayDss.h, vps_displayCtrl.h
Removal	NA	Removal of Mosaic APIs and structures
Replacement	NA	Vps_CropConfig and Vps_PosConfig moved to FVID2 channelNum to chNum numWindows to numWin
SC	NA	Addition to advance configuration parameter as a pointer to Vps_ScConfig structure so that separate IOCTL to set advance config could be avoided
VPS Capture		
Create params	One structure for driver and VIP configuration	Create structure split into generic capture driver structure and VIP configuration structure is part of a separate IOCTL IOCTL_VPS_CAPT_SET_VIP_PARAMS
Instances	2 VIPs x 2 Ports = 4	3 VIPs x 2 Slices x 2 Ports = 12
Addition	NA	Repacker configuration parameters
Replacement	NA	Changes to configuration structures to match the grouping as in VIP hardware

Single Channel Callback	Based on 8 ms timer Asynchronous to capture VSYNC	Synchronous to capture VSYNC
Buffer Capture Modes	Frame drop	Frame drop Last frame repeat Circular frame repeat
Slice based capture	Slice interrupt on Video M3	Slice interrupt on both IPU (local + remote)
Max height/width limitation	Limited Set	Limited set + 3 configurable sizes
	VPS Display	
DSS	NA	Display pipeline is entirely different from HDVPSS. So the interface is entirely different
	VPS VPE M2M	
Interface	vps_m2mDei.h, vps_m2m.h	split into two files vps_m2m.h – Contains generic M2M driver interface vps_m2mVpe.h – Contains interface generic to VPE Based on this the macros and structures are prefixed with Vps_M2m* or Vps_M2mVpe*
Create params	Separate structure for driver and VPE configuration	Create structure split into generic M2M driver structure and VPE configuration structure is part of a separate IOCTL IOCTL_VPS_M2M_SET_VPE_PARAMS. Application has to create the driver and then call this IOCTL for every channel before calling any other APIs
Instance	VPS_M2M_INST_MAIN_DEI_SC1_WB0	VPS_M2M_INST_VPE1
Removal	NA	Removal of DEIHQ related functions, structures and macros

VPS VIP Module Detailed Changes

TI81xx HDVPSS	Tda2xx BSP	Description	Reason for change
FVID2_VPS_CAPT_VIP_DRV	FVID2_VPS_CAPT_VID_DRV	Capture driver ID	BSP is a generic capture driver where as HDVPSS is specific to VIP. Instances will have the VIP instance
VPS_CAPT_INST_VIP_ALL	VPS_CAPT_INST_ALL	Common instance ID for capture driver	Same as above
VPS_CAPT_INST_VIP1_PORTA	VPS_CAPT_VIP_MAKE_INST_ID(VPS_VIP1, VPS_VIP_S0, VPS_VIP_PORTA)	VIP instance ID	In Vayu VIP, we have one VIP having two slices and a slice having two ports. Make the instance ID generation based on above logic instead of hard coded values
VPS_CAPT_INST_VIP1_PORTB	VPS_CAPT_VIP_MAKE_INST_ID(VPS_VIP1, VPS_VIP_S0, VPS_VIP_PORTB)	Same as above	Same as above
VPS_CAPT_INST_VIP2_PORTA	VPS_CAPT_VIP_MAKE_INST_ID(VPS_VIP1, VPS_VIP_S1, VPS_VIP_PORTA)	Same as above	Same as above
VPS_CAPT_INST_VIP2_PORTB	VPS_CAPT_VIP_MAKE_INST_ID(VPS_VIP1, VPS_VIP_S1, VPS_VIP_PORTB)	Same as above	Same as above

VPS_CAPT_FRAME_QUE_LEN_PER_CH_MAX	VPS_CAPT_DEF_QUEUE_LEN_PER_CH	Per channel driver input queue depth	Earlier this was the max the driver could support. Now this represents the default queue depth. Application could change this to any valid values by setting chInQueueLength in create args as driver internally uses dynamic allocation from BSS heap instead of static
VPS_CAPT_SCALAR_ID_DEFAULT	VPS_CAPT_SCALER_ID_DEFAULT	Default scaler ID	Scaler is the right name for VIP SC and hence replaced all occurrence of Scalar with Scaler
Vps_CaptChGetStatusArgs	Vps_CaptChStatusArgs	Channel status args	Consistent naming for structure - not prefix get/set in the structure name
IOCTL_VPS_CAPT_SET_SC_PARAMS	Removed	Set SC params IOCTL	This is part of set VIP params IOCTL
IOCTL_VPS_CAPT_RESET_VIP0	IOCTL_VPS_CAPT_RESET_VIP	Reset VIP0 IOCTL	Since Vayu VIP supports individual resets of ports, these two IOCTLs is merged into one and based on the handle, the driver will reset the necessary port reset
IOCTL_VPS_CAPT_RESET_VIP1	IOCTL_VPS_CAPT_RESET_VIP	Reset VIP1 IOCTL	Same as above
IOCTL_VPS_CAPT_PRINT_ADV_STATISTICS	Removed	Print vip statictics	This is part of IOCTL_VPS_CAPT_GET_CH_STATUS IOCTL. Now application could get this information and print it the way it desires. Application level utility BspUtils_appPrintCaptStatus() is provided to perform this operation
IOCTL_VPS_CAPT_CHECK_OVERFLOW	Removed	Check Vip overflow	Same as above
IOCTL_VPS_CAPT_RESET_AND_RESTART	Removed	Reset VIP and restart	Application should stop/start the driver. Reset is done while re-starting the driver
IOCTL_VPS_CAPT_SET_STORAGE_FMT	Removed	Set buffer storage format IOCTl - field merged or separated	This is part of set VIP params IOCTL
IOCTL_VPS_CAPT_GET_STORAGE_FMT	Removed	Get buffer storage format IOCTl - field merged or separated	This is part of get VIP params IOCTL
IOCTL_VPS_CAPT_SET_VIP_CROP_CFG	Removed	Set VIP crop config	This is part of get VIP params IOCTL
IOCTL_VPS_CAPT_GET_VIP_CROP_CFG	Removed	Get VIP crop config	This is part of get VIP params IOCTL
IOCTL_VPS_CAPT_GET_TILED_MEM_INFO	Removed	Get tiler info IOCTL	Not applicable
IOCTL_VPS_CAPT_SET_TILER_MEM	Removed	Set tiler memory	Not applicable

IOCTL_VPS_CAPT_GET_BACK_TILER_MEM	Removed	Get back tiler memory	Not applicable
createPrms.videoCaptureMode	createPrms.videoIfMode	Capture mode	"Made this a generic Fvid2_VideoIfMode. Hence FVID2_VIFM_SCH_DS_HSYNC_VSYNC or similar macro should be used instead of VPS_CAPT_VIDEO_CAPTURE_MODE_SINGLE_CH_NON_MUX_DISCRETE_SYNC
createPrms.videoIfMode	createPrms.videoIfWidth	Capture interface width	Made this a generic Fvid2_VideoIfWidth. Hence FVID2_VIFW_8BIT should be used instead of VPS_CAPT_VIDEO_IF_MODE_8BIT
createPrms.inDataFormat	Vps_CaptVipParams.inFmt.dataFormat	Input data format	Moved this from create args to VIP params IOCTL
createPrms.periodicCallbackEnable	createPrms.periodicCbEnable	Enable periodic callback mode	Short name, consistent with other variables
createPrms.outStreamInfo	Vps_CaptVipParams.outStreamInfo	Output stream information	Moved this from create args to VIP params IOCTL
createPrms.scParams	Vps_CaptVipParams.scPrms (Pointer)	VIP scaler parameters	Moved this from create args to VIP params IOCTL
createPrms.vipParserPortConfig	Vps_CaptVipParams.vipPortCfg	VIP Port parameters	Moved this from create args to VIP params IOCTL
createPrms.cscConfig	Vps_CaptVipParams.cscCfg	VIP CSC parameters	Moved this from create args to VIP params IOCTL
createPrms.channelNumMap	createPrms.chNumMap	Channel number mapping array	Short name, consistent with other variables
createPrms.inScanFormat	Vps_CaptVipParams.inFmt.scanFormat	Input scan format	Moved this from create args to VIP params IOCTL
Vps_CaptRtParams.captureOutWidth/Height	Vps_CaptRtParams.capturedOutWidth/Height	RT params	Changed verb usage
Vps_CaptOutInfo	Vps_CaptVipOutInfo	VIP output stream information	Prefixed with VIP since this structure is VIP specific and not generic capture driver
VPS_CAPT_MAX_OUT_HEIGHT_UNLIMITED	VPS_VPDMA_MAX_OUT_HEIGHT_UNLIMITED	VPDMA max height	This is not a capture driver parameter. This is a VPDMA parameter valid for VPE/VIP. Similar changes are done for other enums
VPS_CAPT_MAX_OUT_WIDTH_UNLIMITED	VPS_VPDMA_MAX_OUT_WIDTH_UNLIMITED	VPDMA max width	Same as above

Misc

Other Changes

Changes	TI81xx HDVPSS	Tda2xx BSP
Platform		
Removal	NA	Moved board and device dependent functions to respective modules
New	NA	Separate platform ID to SOC ID and platform ID Ex: BSP_PLATFORM_ID_EVM and BSP_PLATFORM_SOC_ID_TDA2SEDX
Devices		
Removal	NA	Moved I2C init params to I2C module as they are independent
Overall Package		
Prefix	PSP	BSP Directory structure changes to use BSP to represent BIOS support package Structures and functions prefixed with Bsp_ instead of Psp_ Macros and enums prefixed with BSP_ instead of PSP_
Structure Init Functions	NA	This sets the structure parameters to default value Application should call init function first and change the default values if required This is added to support backward compatibility in case new members are added to structures

Migration FAQs

- Is the BSP APIs backward compatible to HDVPSS APIs?
  - No. Due to significant changes in architecture, configuration approach and modules, the BSP APIs will not be backward compatible to HDVPSS APIs. But the BSP video drivers follow the same FVID2 queue management APIs and the application flow remains the same.
- Can I use BSP with BIOS 5.xx?
  - No, SYS/BIOS 6.xx must be used.
- What is the performance difference between the two?
  - In profiling the common modules between BSP and HDVPSS on TI81xx, the performance between the two is seen to be comparable.

Article Sources and Contributors

VAYU-BSP MigrationGuide *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?oldid=170485 *Contributors*: SivarajR

Image Sources, Licenses and Contributors

Image:Vayu_BSP_Block_Diagram.PNG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Vayu_BSP_Block_Diagram.PNG *License*: unknown *Contributors*: SivarajR

Image:Vayu_BSP_VPS_Driver_Stack.PNG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:Vayu_BSP_VPS_Driver_Stack.PNG *License*: unknown *Contributors*: SivarajR

Image:VayuBSP_TI81xxHDVPSS_Directory_Structure_Comparison.PNG *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:VayuBSP_TI81xxHDVPSS_Directory_Structure_Comparison.PNG *License*: unknown *Contributors*: SivarajR

Image:VayuBSP_TI81xxHDVPSS_Init_Deinit_Sequence.png *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?title=File:VayuBSP_TI81xxHDVPSS_Init_Deinit_Sequence.png *License*: unknown *Contributors*: SivarajR