TEXAS INSTRUMENTS

DSP/BIOS Resizer Device Driver

User's Manual

April 2009
Release 1.10.00

Platform Software Group

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information 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.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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

Read This First

About This Manual

The API reference guide serves as a software programmer's handbook for working with the Resizer driver module. This reference guide provides necessary information regarding how to use these modules in user systems and applications.

Abbreviations

Table of Abbreviations

Abbreviation	Description
API	Application Programming Interface
DDC	Device Driver Core
IOM	Device Driver Adapter
ISR	Interrupt Service Routine
OS	Operating System
ROM	Read Only Memory
SOC	System On Chip
PALOS	Platform Abstraction Layer operating system
CSL	Chip Support library

Revision History

Date	Author	Comments	Version
November 15,2006	EI3	Created the document	1.0.0
November 20,2006	EI3	Release to TI	Pre-silicon Release 0.3.0
November 29,2006	EI3	Release to TI	Post-silicon Release 0.3.0
December 25, 2006	EI3	Description added for APIs and data structures. Sample application explanation is added	Post-silicon Release 0.3.0
January 2, 2007	EI3	Section 1.6 added.	Post-silicon Release 0.4.0
January 16, 2007	EI3	Bios version changed in Section 1.5	Post-silicon Release 0.4.1
January 19, 2007	EI3	Changes done according to MR DPSP00005799 and Bios Version changed. Description of PSP_RSZ_PIX_FMT_UYVY and PSP_RSZ_PIX_FMT_YUYV in section 3.3 has been modified to give more clear information.	Post-silicon Release 0.5.0
June 22, 2007	Anuj Aggarwal	GA Patch Release 1	1.00.01
June 29,2007	Amit Chatterjee	Modified Release Version	1.00.02
July 18, 2007	EI3	Modified Release Version	1.00.03
November 29, 2007	Sivaraj R	PSP merge package changes - directory structure changes	1.00.04
January 14, 2008	Sivaraj R	PSP_RSZ_IOCTL_SET_SEM_TIMEOUT IOCTL added	1.00.05
January 24, 2008	Sivaraj R	Added TCI file driver initialization illustration and added dependent libraries for building video application	1.00.06
May 21, 2008	Chandan Nath	Updated for adding compiler switches in build options	1.00.07

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This document is an API reference guide on DSP/BIOS Resizer Device Driver for DM6437 SOC.

1.1 H/W S/W Support

This Resizer Device driver has been developed for the DSP/BIOS operating system using the TI supplied Chip Support Library. For more details on the version numbers refer to the release notes in the root of the installation.

1.2 Driver Components

The driver is constituted of following sub components:

RESIZER IOM – Application facing, OS Specific Adaptation of Resizer Device Driver

RESIZER DDC –OS Independent part of Resizer Driver Core

RESIZER CSL –The low-level Resizer h/w abstraction module

System components:

PALOS – DSP/BIOS Abstraction

CSL– Non functional h/w abstraction.

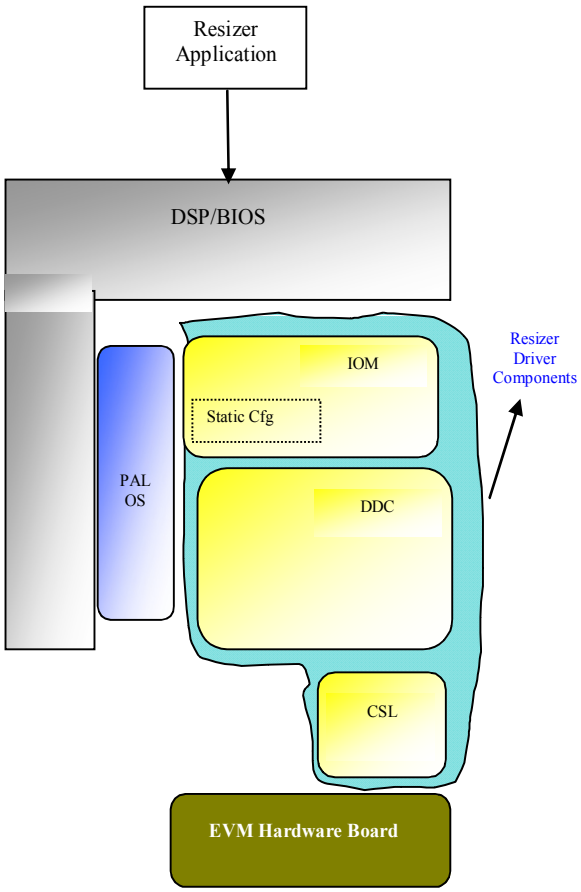


Figure 1 Device Driver Functional Decomposition

1.3 Default Driver Configuration

Resizer driver does not have any default configuration support. Before using the driver, application should configure the driver with valid configurations. In case the driver recognizes invalid configuration it will return the corresponding error.

1.4 Driver Capabilities

The significant driver features are:

- Resizer driver supports input from DDR.
- Resizer driver supports output to DDR.
- Resizer driver works synchronously.
- Resizer driver supports Single pass resizing. The multiple passes for larger buffers should be handled in the application space.
- Resizer driver supports user configurable coefficient programming.
- Resizer driver supports Configuration of Luminance enhancement.
- Resizer driver supports multiple virtual channels with priority support.

1.5 System Requirements

Details about the tools and the BIOS version that the driver is compatible with can be found in the system Release Notes.

1.6 Constraints

The Following is a list of driver and register configuration constraints:

Resizer Output Width

- The output width should be even number of pixels. Minimum output width can be 16 pixels.
- The output width cannot be greater than 1280 pixels if the vertical resizing ratio is between 1/2x to 4x and 640 pixels wide if the vertical resizing ratio is between 1/2x to 1/4x.

Resizer Input Address

- SDRAM or DDRAM source address must be 32-byte aligned.

Resize Ratio

- As actual Resize Ratio is recalculated inside driver, actual value for maximum & minimum scaling factor are little less than 4X & little more than 0.25X respectively.

Chapter 2

INSTALLATION GUIDE

2.1 Component Folder

Upon installing the Resizer driver the following directory structure is found in the driver's directory.

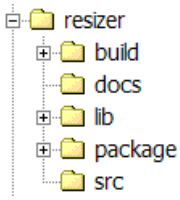


Figure 2 Resizer Driver Directory Structure

This top level resizer folder contains resizer driver psp header file and XDC package files (package.bld, package.xdc and package.xs)

- **build:** This folder contains resizer driver library project file. The generated driver library shall be included in the application where RESIZER driver have to be used.
- **docs:** This folder contains architecture document, datasheet, release notes and user guide.

  Architecture document contains the driver details which can be helpful for the developers as well as consumers to understand the driver design.

  Datasheet gives the idea about the memory consumption by the driver and description of the top level APIs.

  Release Note gives the details about system requirements, steps to Install/Uninstall the package.This document list the known issues of the driver.

  User Guide provides information about how to use the driver. It contains description of sample applications which guide the end user to make their applications using this driver.
- **lib:** This folder contains libraries generated in all the configuration modes(debug, idebug, irelease and release)
- **package:** This folder contains files generated by XDC tool.
- **src:** This folder contains resizer driver source files. It also contains header files that are used by the driver.

2.2 Build

This section describes for each supported target environment, the applicable build options, supported configurations and how selected, the featured capabilities and how enabled, the allowed user customizations for the software to be installed and how the same can be realized.

The component might be delivered to user in different formats:

- Source-less ie., binary executables and object libraries only.
- Source –inclusive.,The entire source code, used to implement the driver is included in the delivered product.
- Source-selective ie., Only a part of the overall source is included. This delivery mechanism might be required either because ; certain parts of the driver require soruce level extensions and/or customization at the user's end or because,specific parts of the driver is exposed to user at the source level to insure user's software development.

When source is included as part of the product delivery, the CCS project file is provided as part of the package. When object format is distributed, the driver header files are part of the "*/drivers/resizer*" folder and the driver library build file is provided in */drivers/resizer/build/dm6437* folder.

2.3 Build Options

This driver does not have any specific build option at the time of writing of this manual.

The build folder contains a CCS project file that builds the driver into a library for debug, idebug, release and irelease mode.

Following compiler switches are used to compile for different options.

- **_DEBUG**
  This is used as a flag to compiler whether to include the debug statements inserted in the code into the final image. This flag helps to build DEBUG image of the program. For RELEASE images this is not passed to the compiler.
- **CHIP_XXXX**
  The CSL layer is written in a common file for all the variants of a SOC. This flag differentiates the variant we are compiling for, for e.g. - CHIP_DM648, and the CSL definitions for that variant appropriately gets defined for register base addresses, num of ports of a peripheral etc.
- **RESIZER_INSTRUMENTATION_ENABLED**
  This flag is passed to the compiler to include the instrumentation code parts into the final image/lib of the program. This helps build the iRelease/iDebug versions of the image/lib with a common code base

Chapter 3

DSP/BIOS RESIZER

This chapter describes the functions, data structures, enumerations and macros for the List module.

3.1 Functions

This section lists the functions available in the PSP module.

3.1.1 GIO_create

GIO_Handle GIO_create	**(String name,** **Int mode,** **Int* status,** **Ptr chanParams,** **GIO_Attrs* attrs)**

This function is called by the application to create the various Resizer channel. The maximum number of channel that can be created is limited to 16.

This function is called by the application to create the RESIZER channel. GIO_create () populates static settings in driver object, formally creates/registers driver entry point with DSP/BIOS,

Parameters:

name	[INOUT] Name of the device to open
mode	[INOUT] Mode in which device is to be opened.
status	[OUT] Address to which driver returns status.
chanParams	[IN] segment id for Memory allocation (zero for external allocation)
attrs	[IN] pointer to GIO_attrs structure

Returns:
GIO_Handle – if the operation is successful
NULL – if the operation is failed

Example:

```
GIO_Handle rszHandle;
Int SegId=0;
Int giostatus;
rszHandle = GIO_create(deviceName, IOM_INOUT, &giostatus, (void
*)&segId,NULL);
```

```
DeviceName is passed as resizer. Instance is declared in the
configuration file.
```

3.1.2 GIO_delete

Int GIO_delete	**(GIO_Handle gioChan)**

This function deletes the driver channel and undoes the memory allocation made by the GIO_create

Parameters:

gioChan	[IN] GIO Handle for the channel

Returns:
IOM_EBADARGS – If the parameters passed are not correct or the channel has never been created

Example:

```
GIO_Handle rszHandle;

GIO_delete(rszHandle);
```

3.1.3 GIO_control

PSP_Result GIO_control	**(GIO_Handle *gioChan*, Int *cmd*, Ptr *cmdArg*)**

This function handles the IOCTLs for the resizer driver.

Parameters:

gioChan	[IN] GIO Handle for the channel
cmd	[IN] IOCTL Command
cmdArg	[INOUT] Argument for the IOCTL

Returns:
IOM_COMPLETED if successful or else suitable error code is given.
IOM_EBADARGS– *gioChan* is not valid or if the state is not appropriate or the argument passed is not appropriate
IOM_EBADMODE– if the cmdArg is not appropriate or if the mode is not supported

Example:

```
GIO_Handle rszHandle;
static PSP_RSZparams params =
```

```
{
    320,                                /* Input image width */
    240,                                /* Input image height */
    320*(2u),                           /* Input image pitch */
    PSP_RSZ_INTYPE_YCBCR422_16BIT,      /* Input image type */
    0,                                  /* HorizonatlHorizontal starting
pixel */
    0,                                  /* Vertical starting pixel */
    0,                                  /* Chroma position */
    PSP_RSZ_PIX_FMT_YUYV,               /* Image format */
    640,                                /* Output image width */
    480,                                /* Output image height */
    640*(2u),                           /* Output image pitch */
    0,                                  /* Horizontal starting phase */
    0,                                  /* Vertical starting phase */
    hcoeffs,                            /* Horizontal filter coefficients*/
    vcoeffs,                            /* Vertical filter coefficients */
   PSP_RSZ_YENH_DISABLE                 /* Luma enhancement options */
};

/* configure the logic channel */
GIO_control(rszHandle, PSP_RSZ_IOCTL_SET_PARAMS, &params);

The various the IOCTL command names can be got from Section 3.3
```

3.2 Data Structures Configuration defines

The file **psp_resizer.h** has the **_PSP_RSZyenhParams** data structure that is passed for configuration of luma enhancement parameters in the GIO_control .The parameters of the structure are explained below.

1) Luma Enhancement Data Structure

Parameter	Description
type	Represents luma enable or disable. If luma type is disabled than gain slop and core values are ignored.
gain	Represents gain on the edges
slop	Represents slop
core	Represents core value

The file **psp_resizer.h** has the **_PSP_RSZparams** data structure that is passed for configuration of resizer parameters in the GIO_control. The parameters are explained below:

2) Resizer Configuration parameters Data Structure

Parameter	Description
inHsize	Represents input frame horizontal size

inVsize	Represents input frame vertical size.
inPitch	Represents offset between two rows of input frame. The pitch is represented in terms of bytes per pixel. For YUV interleaved, bytes per pixel are 2 and for color separate, bytes per pixel is 1.
Inptyp	For determining 16 bits/pixel (YUV Interleaved) or 8 bits/pixel (Color separate) data
vertStartingPixel	For specifying vertical starting pixel in input. If the input is from DDR than this value is always zero
horzStartingPixel	For specifying horizontal starting pixel in input
Cbilin	Defined, filter with luma or bi-linear Interpolation
Pixfmt	Defined, UYVY or YUYV image format
outHsize	Represents output frame horizontal size.
outVsize	Represents output frame vertical size. For resize ratio ½ x to 4x outVsize cannot be greater than 1280 and for resize ratio ½ x to ¼ x outVsize cannot be greater than 640.
outPitch	Represents offset between two rows of output frame. The pitch is represented in terms of bytes per pixel. For YUV interleaved bytes per pixel are 2 and for color separate bytes per pixel is 1.
Hstph	For specifying horizontal starting phase
Vstph	For specifying vertical starting phase
hfiltCoeffs	Horizontal filter coefficients. If the resize ratio is between ½ x to 4x than use 4-tap 7-phase filter coefficients and if ratio is between ½ x to ¼ x use 7-tap 4-phase filter coefficients. The filter coefficients are to be configured from application and user may have to fine tune these coefficients for different images.
vfiltCoeffs	Vertical filter coefficients. . If the resize ratio is between ½ x to 4x than use 4-tap 7-phase filter coefficients and if ratio is between ½ x to ¼ x use 7-tap 4-phase filter coefficients. The filter coefficients are to be configured from application and user may have to fine tune these coefficients for different images.
yenhParams	Represents luma enhancement parameters

The file **psp_resizer.h** has the **_PSP_RSZresize** data structure that is passed for configuration of parameters required during resizing. The structure is passed via GIO_control. The parameters are explained below:

3) Resizing Data Structure

Parameter	Description
inBuf	Represents input frame starting address. It should 32-byte aligned.
outBuf	Represents output frame starting address. It should 32-byte aligned.
inBufSize	Represents input buffer size. Buffer size is calculated as inHsize *inPitch.

outBufSize	Represents output buffer size. Buffer size is calculated as outHsize *outPitch.

The file **psp_resizer.h** has the **_PSP_RSZstatus** data structure that is passed to GIO_control to get the current status of hardware, channel and to get the current input source. The parameters are explained below:

4) Resizer StatusTable Configuration Data Structure

Parameter	Description
chanBusy	1: channel is busy, 0: channel is not busy
hwBusy	1: hardware is busy, 0: hardware is not busy
Src	Defined, can be either DDR or CCDC/PREVIEWER.

The file **psp_resizer.h** has the **_PSP_RSZpriority** data structure that is passed to GIO_control function to get and set priority of the driver. The parameters are explained below:

5) Resizer priority settings Configuration Data Structure

Parameter	Description
priority	To set the priority of configuration channel.0 to 5, with 5 as the highest priority

The file **psp_resizer.h** has the **_PSP_RSZcropsize** data structure that is passed to GIO_control function to get the cropping information. The parameters are explained below:

6) Resizer priority settings Configuration Data Structure

Parameter	Description
hcrop	Represents number of pixels per line cropped in output image
vcrop	Represents number of lines cropped in output image

3.3 Symbolic Constants and Enumerated Data Types

This section lists the enumerations available in the PSP module.

Group or Enumeration Class	Symbolic Constant Name	Description or Evaluation
Macro	PSP_RESIZER_NUM_CHANNELS	Maximum number of channels supported by driver. The maximum number is restricted to 16
Macro	PSP_RSZ_NUM_INSTANCES	Maximum number of Driver instance supported by driver. The maximum number is

		restricted to 1.
Macro	PSP_RSZ_CBILIN_DISABLE	Indicates that luminance processing is disable. Its value is represented by 0. This macro is used to assign value to cbilin field of _PSP_RSZparams
Macro	PSP_RSZ_CBILIN_ENABLE	Indicates that luminance processing is enable. Its value is represented by 1. This macro is used to assign value to cbilin field of _PSP_RSZparams
Macro	PSP_RSZ_PIX_FMT_UYVY	Indicates the position of luma and chroma component. If this macro is used than it means that chroma component data is in msb and luma is in lsb.So when the input format is UYVY or VYUY than keep assign this value to Pixfmt field of _PSP_RSZparams
Macro	PSP_RSZ_PIX_FMT_YUYV	Indicates the position of luma and chroma component. If this macrp is used than it means that luma data is in msb and chroma is in lsb. So when the input format in YUYV or YVYU than assign this value to Pixfmt field of _PSP_RSZparams
Macro	PSP_RSZ_PIX_FMT_PLANAR	Indicates that input image is 8 bit color separated. Its value is represented by 2. This macro is used to assign value to Pixfmt field of PSP_RSZparams
Macro	PSP_RSZ_YENH_DISABLE	Indicates the luma enhancement algorithm is disabled. Its value is represented by 0.This

		macro is used to assign value to type of _PSP_RSZyenhParams
Macro	PSP_RSZ_YENH_3TAP_HPF	Indicates the luma enhancement algorithm used is 3 Tap Its value is represented by 1.This macro is used to assign value to type of _PSP_RSZyenhParam
Macro	PSP_RSZ_YENH_5TAP_HPF	Indicates the luma enhancement algorithm used is 5 Tap Its value is represented by 2. This macro is used to assign value to type of _PSP_RSZyenhParam
Macro	PSP_RSZ_INTYPE_YCBCR422_16BIT	Indicates input image is 16 bit color interleaved. Its value is represented by 0 .This macro is used to assign value to Inptyp field of _PSP_RSZparams
Macro	PSP_RSZ_INTYPE_PLANAR_8BIT	Indicates input image is 8 bit color separate. Its value is represented by 0. This macro is used to assign value to Inptyp field of _PSP_RSZparams
Enumeration	PSP_RSZ_IOCTL_SET_PARAMS	Control Command to set resizer channel configuration parameters. This command will not write to hardwareconfigure hardware registers, It will just store the parameters in the channel configuration structure inside driver. _PSP_RSZparams structure is passed in GIO_control for this control command;
Enumeration	PSP_RSZ_IOCTL_GET_PARAMS	Control Command to get channel configuration

		parameters. _PSP_RSZparams structure is passed in GIO_control for this control command;
Enumeration	PSP_RSZ_IOCTL_RESIZE	Control Command for resizing. This command will configure resizer hardware registers with appropriate channel parameters and perform resizing according to priority. The resizing for larger buffer should be done in 2two pass from application. _PSP_RSZresize structure is passed in GIO_control for this control command;
Enumeration	PSP_RSZ_IOCTL_SET_PRIORITY	Control Command to set channel priority. _PSP_RSZpriority structure is passed in GIO_control for this control command;
Enumeration	PSP_RSZ_IOCTL_GET_PRIORITY	Control Command to get channel priority parameters. _PSP_RSZpriority structure is passed in GIO_control for this control command;
Enumeration	PSP_RSZ_IOCTL_GET_STATUS	Control Command to get status of hardware and channel. _PSP_RSZstatus structure is passed in GIO_control for this control command;
Enumeration	PSP_RSZ_IOCTL_GET_CROPSIZE	Control Command to get cropsize. _PSP_RSZcropsize structure is passed in GIO_control for this control command;
Enumeration	PSP_RSZ_IOCTL_SET_EXP	Control Command to set delay between two

		consecutive accesses to write buffers. Address of the value that is to be configured is passed in GIO_control for this control command.
Enumeration	PSP_RSZ_IOCTL_SET_SEM_TIMEOUT	This control command is used to set the timeout values used in semaphore operation in the driver - values are in ms.
Enumeration	PSP_RSZ_IOCTL_SET_RESIZE_CALLBACK	This control command is used to set (register) a callback with the resize driver to notify the completion of resize operation. This callback if used then the command to be resize is as described next (Please refer to the sample application section for more details on usage)
Enumeration	PSP_RSZ_IOCTL_RESIZE_ASYNC	This control command is used to start a resize operation. This command does not acquire any semaphores in the path and hence could be used by users who do not wish to be put to “sleep” while the resizing is in progress. (Please refer to the sample application section for more details on usage)

CHAPTER 4

EXAMPLE APPLICATIONS

This section describes the example applications that are included in the package. These sample applications can be run as is for quick demonstration, but the user will benefit most by using these samples as reference source code in developing new applications.

4.1 Writing Applications for Resizer

This section provides guidance to user for writing their own application for Resizer driver

4.1.1. File Inclusion

To write sample application user has to include following header files in the application:

1. **psp_resizer.h**

   This file contains the interfaces, data types and symbolic definitions that are needed by the application to utilizes the services of Resizer device driver.

4.1.2. Driver Initialization

To use the Resizer device driver, a device entry must be added and configured in the DSP/BIOS configuration tool.

To have Resizer device driver included in the application, corresponding TCI file have to be included in BIOS TCF i.e. "*dm6437_resizer.tci*" must be included in BIOS TCF file of the application. This file can be found in video sample directory.

The Resizer driver initialization in BIOS TCF looks like the following:

```
bios.UDEV.create("resizer");
bios.UDEV.instance("resizer").fxnTable = prog.extern("RSZMD_FXNS");
bios.UDEV.instance("resizer").fxnTableType = "IOM_Fxns";
```

4.1.3. Dependent Projects/Libraries

Following are the dependent libraries/projects to successfully build video application

- Resizer
- PAL_OS
- SoC specific PAL_SYS

4.1.4 Pragma directives used in the Applications

- DATA_ALIGN
  - Any buffer used for storing/retrieving data should be cache aligned at 128 bytes, since they write/read, to/from SDRAM/DDRAM.
  - The CCDC and OSD source and destination addresses should always be on 32-byte alignment.

4.2 The Resizer Downscale Sample Application

4.2.1. Introduction

This psp_bios_resz_st_downscale_example.pjt application demonstrates downscaling feature of Resizer. It resizes 640x480 image to 320x240 image.

4.2.2. Building the Application

The sample application project file is located in the *<root>\packages\ti\sdo\pspdrivers\system\dm6437\bios\evmDM6437\video\sample\build\resizer* folder. The sample can be rebuilt directly from this project file using Code Composer studio.

4.2.3. Loading the Application

The sample application is loaded and executed via Code composed studio. It is recommended to reset the board before loading Code Composer

4.3 The Resizer Upscale Sample Application

4.3.1. Introduction

This psp_bios_resz_st_upscale_example.pjt application demonstrates upscaling feature of Resizer. It resizes 320x240 image to 640x480 image.

4.3.2. Building the Application

The sample application project file is located in the *<root>\packages\ti\sdo\pspdrivers\system\dm6437\bios\evmDM6437\video\sample\build\resizer* folder. The sample can be rebuilt directly from this project file using Code Composer studio.

4.3.3. Loading the Application

The sample application is loaded and executed via Code composed studio. It is recommended to reset the board before loading Code Composer

4.4 The Resizer Multipass Sample Application

4.4.1. Introduction

This psp_bios_resz_st_multipass_example.pjt application application demonstrates multipass functionality. It downscales 1280x640 image to 128x64 image.

4.4.2. Building the Application

The sample application project file is located in the *<root>\packages\ti\sdo\pspdrivers\system\dm6437\bios\evmDM6437\video\sample\build\resizer* folder. The sample can be rebuilt directly from this project file using Code Composer studio.

4.4.3. Loading the Application

The sample application is loaded and executed via Code composed studio. It is recommended to reset the board before loading Code Composer

4.5. The Resizer Multipass Sample Application (with callback support)

4.5.1. Introduction

The resize driver provides the PSP_RSZ_IOCTL_RESIZE ioctl command to start the resize operation on the image given in input buffer provided as part of the command argument. However, this resize path contains semaphores and hence mandates that the caller is in the task context. This may not be desirable for some users. To circumvent this problem, new resize IOCTL command path for resize has been provided via PSP_RSZ_IOCTL_RESIZE_ASYNC. The usage of this is detailed below.

1. register a callback with driver via PSP_RSZ_IOCTL_SET_RESIZE_CALLBACK
    a. The argument for this command is the pointer to the structure PSP_RSZcallback containing the application callback function and the application callback argument.
2. Set the resize parameters via PSP_RSZ_IOCTL_SET_PARAMS
3. Start resize via PSP_RSZ_IOCTL_RESIZE_ASYNC.
4. The call to PSP_RSZ_IOCTL_RESIZE_ASYNC returns immediately with IOM_PENDING in case of success, and proper error code in case of errors
5. The user should wait for the completion of notification before submitting next resize request. In the sample application, after the completion of resize in callback the next request is submitted. User may decide on the mode of waiting.

Note:

1. Only one channel per instance is possible in this mode
2. This mode is automatically selected, via PSP_RSZ_IOCTL_SET_RESIZE_CALLBACK, and after this, only this mode of resize only is possible until the device (instance) lifetime.
3. Need to call PSP_RSZ_IOCTL_SET_RESIZE_CALLBACK only once.

The sample application project psp_bios_resz_st_multipass_callback_example.pjt demonstrates multipass functionality.

4.5.2. Building the Application

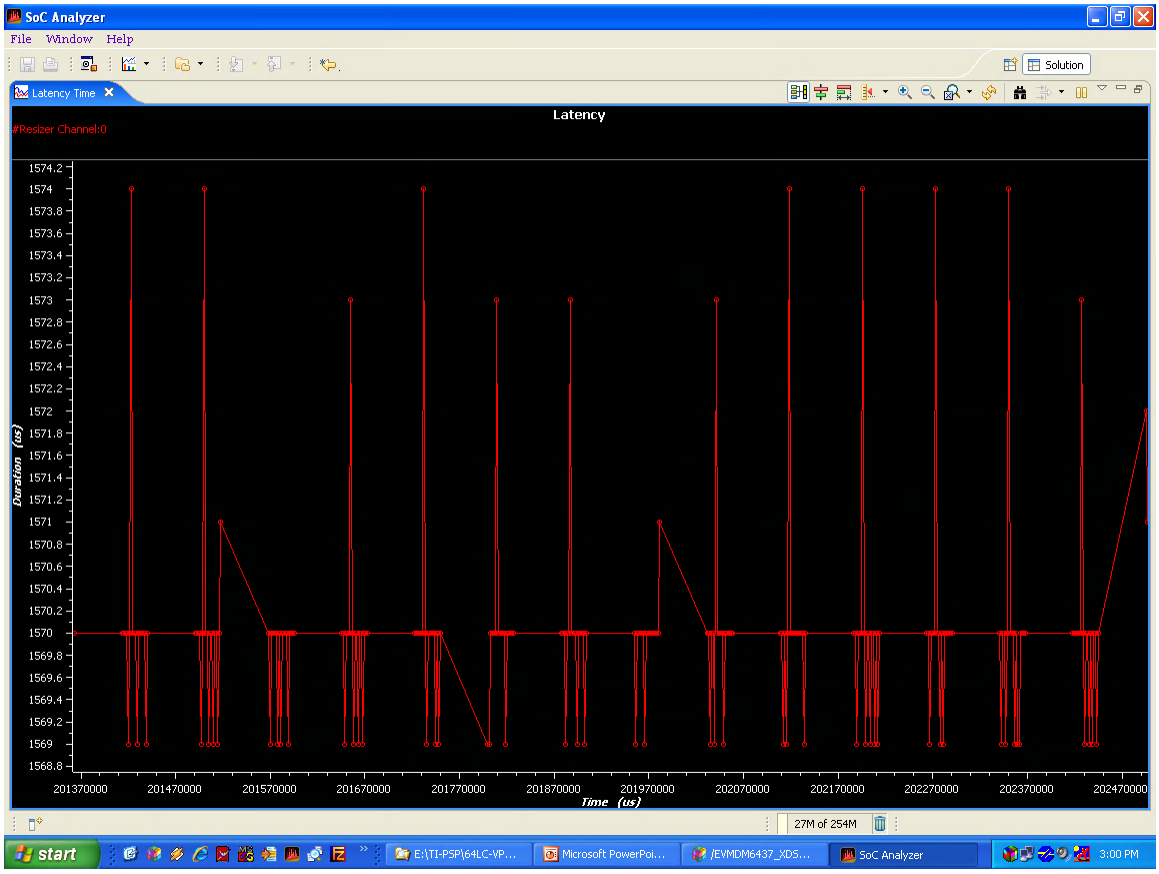
The sample application project file is located in the *<root>\packages\ti\sdo\pspdrivers\system\dm6437\bios\evmDM6437\video\sample\build\resizer* folder. The sample can be rebuilt directly from this project file using Code Composer studio.

4.5.3. Loading the Application

The sample application is loaded and executed via Code composed studio. It is recommended to reset the board before loading Code Composer

Chapter 5

RESIZER PERFORMANCE RESULTS



- latency of RESIZER driver in interrupt mode

Note: The Driver Performance Characteristics can be included once testing is done on DM6437/C6424 SOC. The graph is taken for Resizer operating in DMA mode.