TII DM6437 VPSS Drivers
Resizer API Specifications

Release 1.10.00

January 14, 2008

Table of Contents

Revision History 3
1. Overview 4
1.1 Purpose and Scope 4
1.2 Names and Terminology 4
1.3 Architecture 5
1.4 Critical Features and Implementation 6
2. Application Level APIs 7
2.1 GIO_CREATE 7
2.2 GIO_DELETE 8
2.3 GIO_CONTROL 8
3. RSZ module Controls 9
3.1 PSP_RSZ_IOCTL_SET_PARAMS 9
3.2 PSP_RSZ_IOCTL_GET_PARAMS 10
3.3 PSP_RSZ_IOCTL_RESIZE 10
3.4 PSP_RSZ_IOCTL_GET_STATUS 11
3.5 PSP_RSZ_IOCTL_SET_PRIORITY 12
3.6 PSP_RSZ_IOCTL_GET_PRIORITY 12
3.7 PSP_RSZ_IOCTL_GET_CROPSIZE 13
3.8 PSP_RSZ_IOCTL_SET_EXP 13
3.9 PSP_RSZ_IOCTL_SET_SEM_TIMEOUT 14
Usage Examples 15
3.10 Registration of resizer driver 15
3.11 Driver open and close 15
3.12 Setup resizing parameters 15
3.13 Perform the resizing operation 16

Revision History

Date	Version	Changes	Author
October 9, 2006	Draft 0.01	Created	EI3
October 10, 2006	Issue 1.00	Updated as per technical review comments	EI3
October 19, 2006		Added PSP_RSZ_IOCTL_SET_EXP	EI3
November 19, 2006	Issue 1.01	Issued to TII	EI3
November 19, 2006	Pre-silicon Release 0.3.0	Release to TI	EI3
December 4, 2006	Post-silicon Release 0.3.0	Release to TI	EI3
April 21, 2007	Post-silicon Release 0.7.0	Release to TI	EI3
May 8, 2007	GA Release 1.0.0	Release to TI	EI3
June 22, 2007	1.00.01	GA Patch Release 1	Anuj Aggarwal
June 29,2007	1.00.02	Modified Release Version	Amit Chatterjee
July 18, 2007	1.00.03	Modified Release Version	EI3
November 29, 2007	1.00.04	PSP merge package changes - directory structure changes	Sivaraj R
January 14, 2008	1.00.05	PSP_RSZ_IOCTL_SET_SEM_TIMEOUT IOCTL added	Sivaraj R

1. Overview

Purpose and Scope

This document provides APIs for the proposed driver for the Resizer on DM6437 family SOCs. The APIs are based on the requirement document that has been agreed upon by the Catalog/EEE team, the PSP team and e-Infochips.

The intention of this document is to provide guidelines on how the driver should behave from application point of view. However, the actual design of the driver is not covered.

Names and Terminology

The module name of the Resizer driver shall be resizer. Hence the name of the top level files which will directly interact with application shall be “dda_rszIOM.c” and “dda_rszIOM.h”. This above files will interact with the dda_rsz.c. Thereafter the dda_rsz.c will interact with the ddc_rsz.c. Finally, the files related to hardware block is referred to as the llc_rsz.c and llc_rsz.h

Architecture

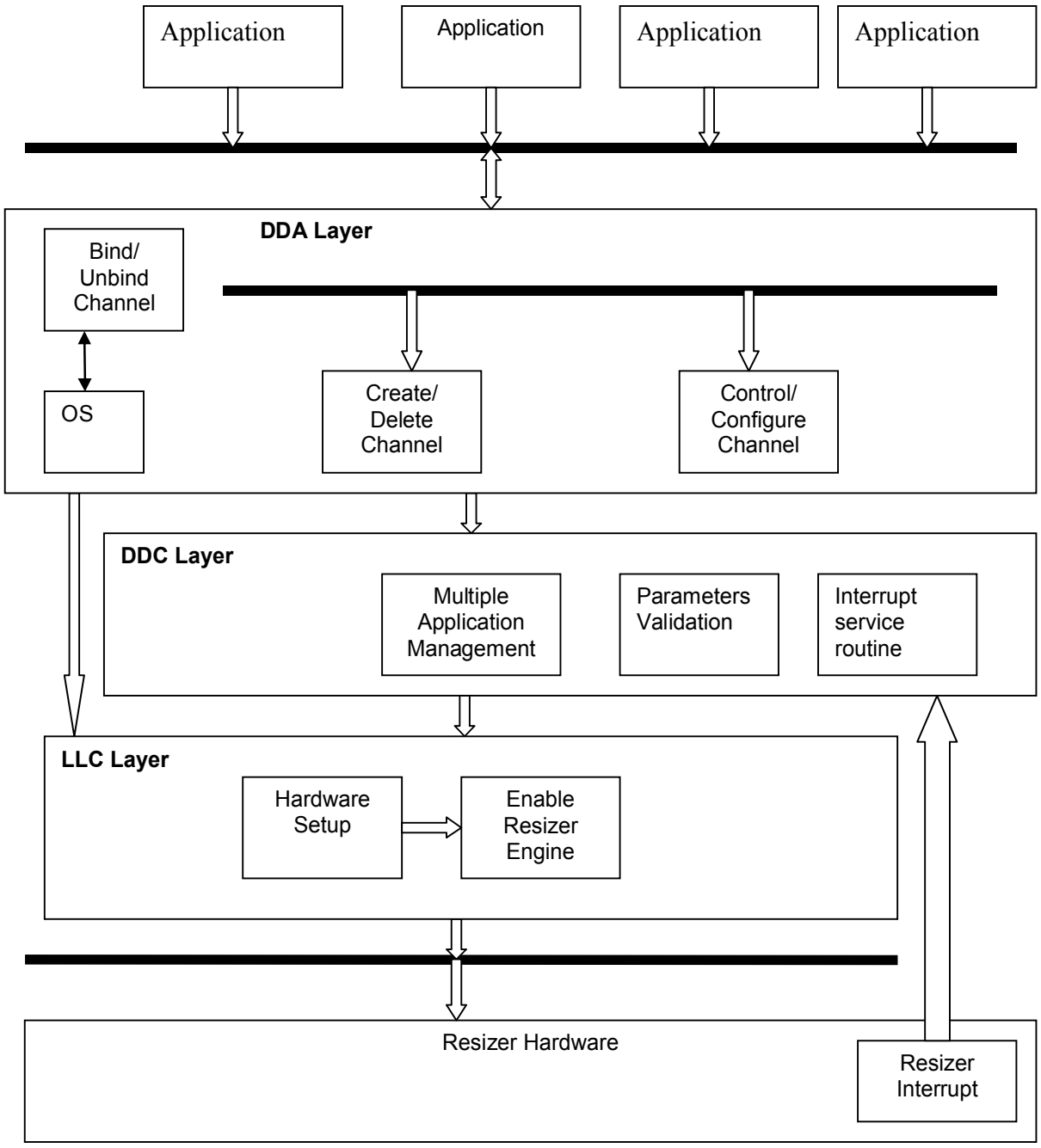

Figure 1. Top-Level Block Diagram of Resizer

The RSZ driver is sub-divided into following horizontal layers:

DDA Layer
This layer handles all OS level driver API implementations. This layer is OS centric and hardware agnostic. It does following functionalities:

- Registration/Unregistration
- Open/Close
- Various controls to configure HW

DDC Layer
This layer is mainly responsible for handling multiple channel support. It validates the parameters send by the application. It also handles the ISR. The Resizer interrupt is generated after the completion of resizing.

LLC Layer
This layer is responsible for the actual configuration of the resizer hardware by writing to the resizer MMRs. It is pretty much OS agnostic and hardware centric. The layer enables the resizer engine after the specific hardware is configured.

Critical Features and Implementation

First, the driver shall support multiple logic channels with only one available hardware Resizer block. It is thus very important that each channel maintains its own parameter settings.

2. Application Level APIs

The following GIO Class driver API shall be supported by the Resizer driver.

- GIO_create()
- GIO_delete()
- GIO_control()

GIO_CREATE

Synopsis
GIO_Handle GIO_create(String name, int mode, int Status, Ptr optargs, GIO_Attrs * attrs);

Arguments
Name
The name argument is the name specified for the device when it was created in the configuration or at runtime. It is used to find a matching name in the device table. The name generally will be "/resizer"
Mode
The mode argument specifies the mode in which the device is to be opened. This may be IOM_INPUT, IOM_OUTPUT, or IOM_INOUT.
Status
If the status parameter is non-NULL, a status value is placed at the address specified by the status param.
Optargs
The optargs parameter is a pointer that may be used to pass device or domain-specific arguments to the mini-driver. The contents at the specified address are interpreted by the mini-driver in a device-specific manner.The memory segment id for memory allocation is passed via this parameter.
Attrs
The attrs parameter is a pointer to a structure of type GIO_Attrs.

Description
Open a logical channel. Multiple open shall be supported by the Resizer driver. The purpose is to support the scenario when the Resizer is used for both video front end pre-processing and back end post-processing. For example, D1 -> CIF for pre-processing and CIF -> D1 for post-processing.

The GIO_Attrs structure is as shown below

```
typedef struct GIO_Attrs
{
        Int nPackets;     /* number of I/O packets */
        Uns timeout;      /* for blocking calls    */
} GIO_Attrs;
```

Return Value
It returns the handle of type GIO_Handle on successful opening of a device. It returns NULL if the device could not be opened.

GIO_DELETE

Synopsis
int GIO_delete(GIO_Handle gioChan);

Arguments
gioChan
Handle to device instance to be closed

Description
Close the logic channel associated with gioChan.

Return Value
IOM_COMPLETED on success, or negative value if an error occurred

GIO_CONTROL

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, int args);

GioChan
Handle to an instance of the device
cmd
Control functionality to perform
args
Data structure to pass control information

Description
An application calls GIO_control to configure or perform control functionality on the communication channel. Macros and defines specifying RSZ control requests are located in the psp_resizer.h header file

Return Value
IOM_COMPLETED on success and negative value if error.

3. RSZ module Controls

PSP_RSZ_IOCTL_SET_PARAMS

Name
PSP_RSZ_IOCTL_SET_PARAMS – set the Resizer hardware parameters associated with this channel

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_RSZparams *argp);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_SET_PARAMS
argp
pointer to the PSP_RSZparams structure

Description
This ioctl is used to set the parameters of the Resizer hardware, including input and output image size, horizontal and vertical poly-phase filter coefficients, luma enchance filter coefficients ,starting horizontal and vertical pixels etc. The parameters are file descriptor specific, i.e., each gioChan returned by the GIO_create() function has its own set of parameters. Only input and output sizes are needed from the application. The driver is responsible for calculating the actual register settings based on those values.

The rsz_params and its fields as defined in the psp_resizer.h header file as shown below.

```
typedef struct _PSP_RSZparams{
    int inHsize;
    int inVsize;
    int inptyp;          // for determining 16 bit or 8 bit data
    int inPitch;
    int vertStartingPixel; // for specifying vertical starting pixel in input
    int horzStartingPixel; // for specfying horizontal starting pixel in input
    int cbilin;       /* # defined, filter with luma or bi-linear interpolation  */
    int pixfmt;     /* # defined, UYVY or YUYV */
    int outHsize;
    int outVsize;
    int outPitch;
    int vstph;  // Determines the starting vertical phase
    int hstph;  // Determines the starting horizontal phase
    short hfiltCoeffs[32];
    short vfiltCoeffs[32];
    PSP_RSZyenhParams yenhParams;
} PSP_RSZparams;

/* The field in_type has 2 possible values */
#define PSP_RSZ_INTYPE_YCBCR422_16BIT       0
#define PSP_RSZ_INTYPE_PLANAR_8BIT        1

/* The field cbilin has 2 possible values */
```

```
#define PSP_RSZ_CBILIN_DISABLE          0    /* same as luminance processing */
#define PSP_RSZ_CBILIN_ENABLE 1    /* enable bi-linear processing       */

/* The field pix_fmt has 2 possible values */
#define PSP_RSZ_PIX_FMT_UYVY 0    /* cb:y:cr:y */
#define PSP_RSZ_PIX_FMT_YUYV 1    /* y:cb:y:cr */
#define PSP_RSZ_PIX_FMT_PLANAR        2    /* 8-bit planar input */

 typedef struct _PSP_RSZyenhParams{
      int type;
      unsigned char gain;
      unsigned char slop;
      unsigned char core;
 } PSP_RSZyenhParams;

 /* The field type has 3 possible values */
 #define      PSP_RSZ_YENH_DISABLE     0
 #define      PSP_RSZ_YENH_3TAP_HPF   1
 #define      PSP_RSZ_YENH_5TAP_HPF   2
```

Return Value
IOM_COMPLETED on success and negative value if error.

PSP_RSZ_IOCTL_GET_PARAMS

Name
PSP_RSZ_IOCTL_GET_PARAMS – get the Resizer hardware parameters associated with this logic channel.

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_RSZparams*argp);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_GET_PARAMS
argp
pointer to the _PSP_RSZparams structure

Description
This ioctl is used to get the Resizer hardware settings associated with the current logic channel represented by fd.

Return Value
IOM_COMPLETED on success and negative value if error.

PSP_RSZ_IOCTL_RESIZE

Name
PSP_RSZ_IOCTL_RESIZE– submit a resizing task to the logic channel associated with fd

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_RSZresize*argp);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_RESIZE
argp
pointer to the _PSP_RSZresize structure

Description
This ioctl is used to submit a resizing task for the specific logic channel using channel specific parameters set by PSP_RSZ_IOCTL_S_PARAM.

This ioctl submit a resizing task specified by the _PSP_RSZresize structure.

The _PSP_RSZresize structure is defined in psp_resizer.h header file as shown below:

```
typedef struct _PSP_RSZresize {
        void *  inBuf;  /* address of the input buffer */
        void *  outBuf; /* address of the output buffer */
        int inBufSize; /* input buffer size */
        int outBufSize; /* output buffer size */
} PSP_RSZresize;
```

Return Value
IOM_COMPLETED on success and negative value if error.

PSP_RSZ_IOCTL_GET_STATUS

Name
PSP_RSZ_IOCTL_GET_STATUS – get the current status of the hardware

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_RSZstatus *argp);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_GET_STATUS
argp
pointer to the _PSP_RSZstatus structure

Description
This ioctl is used to check the current status of the Resizer hardware.
The channel busy field of the structure below indicates that the current channel is into processing.
The hardware busy indicates that currently one channel is being served by the hardware.

The _PSP_RSZstatus structure is defined in psp_resizer.h header file as shown below:

```
typedef struct _PSP_RSZstatus {
        int chanBusy;   /* 1: channel is busy, 0: channel is not busy */
        int hwBusy;     /*  1: hardware is busy, 0: hardware is not busy */
        int src;  /* # defined, can be either SD-RAM or CCDC/PREVIEWER */
} PSP_RSZstatus;
```

```
/* the src field in _PSP_RSZstatus structure has 2 possible values */

/* currently the resizer can only take input from the RAM.     */
#define PSP_RSZ_SRC_CCDC_PREVIEWER   0
#define PSP_RSZ_SRC_RAM                                 1
```

Return Value
IOM_COMPLETED on success and negative value if error.

PSP_RSZ_IOCTL_SET_PRIORITY

Name
PSP_RSZ_IOCTL_SET_PRIORITY – set the priority of the logic channel identified by fd.

Synopsis
int GIO_conntrol(GIO_Handle gioChan, int cmd, PSP_RSZpriority *argp);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_SET_PRIORITY
argp
pointer to the _PSP_RSZpriority structure

Description
This ioctl is used to set the priority of the current logic channel. If multiple resizing tasks from multiple logic channels are currently pending, the task associated with the highest priority logic channel will be executed first.

The _PSP_RSZpriority structure is defined in psp_resizer.h header file as shown below:

```
typedef struct _PSP_RSZpriority {
        int priority;   /* 0=>5, with 5 the highest priority */
} PSP_RSZpriority;
```

Return Value
IOM_COMPLETED on success and negative value if error.

PSP_RSZ_IOCTL_GET_PRIORITY

Name
PSP_RSZ_IOCTL_GET_PRIORITY – get the current priority setting of the logic channel.

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_RSZpriority *argp);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_GET_PRIORITY

argp
pointer to the _PSP_RSZpriority structure

Description
This ioctl is used to get the priority of the current logic channel

Return Value
IOM_COMPLETED on success and negative value if an error occurred

PSP_RSZ_IOCTL_GET_CROPSIZE

Name
PSP_RSZ_IOCTL_GET_CROPSIZE – get the cropping size of output image.

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_RSZcropsize *argp);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_G_CROPSIZE
argp
pointer to the _PSP_RSZcropsize structure

Description
This ioctl is used to get the size reduction in the output image compared to input image in terms of number of pixels per line and number of lines depending on features enabled.

The _PSP_RSZcropsize structure is defined in psp_resizer.h header file as shown below:

```
typedef struct _PSP_RSZcropsize  {
        unsigned int hcrop ;      /*  number of pixels per line cropped in output image */
        unsigned int vcrop;       /*  number of lines cropped in output image */
} PSP_RSZcropsize;
```

Return Value
IOM_COMPLETED on success and negative value if error.

PSP_RSZ_IOCTL_SET_EXP

Name
PSP_RSZ_IOCTL_SET_EXP – set the delay time for read request.

Synopsis
int GIO_conntrol(GIO_Handle gioChan, int cmd, int *);

Arguments
gioChan
Handle to an instance of the device
request
PSP_RSZ_IOCTL_SET_EXP

argp

pointer to the delay time value.

Description

Delay to allow between consecutive read requests from the Resizer module. Units are of 32 VPSS clock cycles (153/198 MHz in Normal/Turbo modes). The delay is RESZ_EXP*32 VPSS clock cycles.

Since VPSS DMA priority is typically set to the highest in the SoC for real-time requirements, this is for spreading any non real-time reads with respect to the other traffic in the system to minimize the potential of locking out other requests for the duration of a frame being read from DDR/SDR.

Return Value

IOM_COMPLETED on success and negative value if error.

PSP_RSZ_IOCTL_SET_SEM_TIMEOUT

Name

PSP_RSZ_IOCTL_SET_SEM_TIMEOUT – set the timeout values used in semaphore operation in the driver. Values are in milliseconds.

Synopsis

int GIO_control(GIO_Handle gioChan, int cmd, Int32 *timeout);

Arguments

gioChan

Handle to an instance of the device

Request

PSP_RSZ_IOCTL_SET_SEM_TIMEOUT

Argp

Pointer to Int32 – timeout in milliseconds; -1 should be provided for infinite timeout.

Description

This control command is used to set the timeout values used in semaphore operation in the driver associated with the current logic channel represented by gioChan.

Return Value

IOM_COMPLETED on success and negative value if error.

Usage Examples

This section provides some example code showing how to use the RSZ module.

Registration of resizer driver

To configure a mini-driver in the DSP/BIOS Configuration Tool, follow these steps:

1. Create a new device object by right-clicking on User-Defined Devices (in the Input/Output tree) and selecting Insert UDEV from the pop-up menu.

2. Rename the object as resizer.

3. Right-click on the UDEV object you created and choose Properties.

4. In the Properties dialog, specify the Initialize function name, name of the function table and function table type .See below

The Function table is as below:-

```
IOM_Fxns RSZMD_FXNS =
{
&RSZ_mdBindDev,
&RSZ_mdUnBindDev,
&RSZ_mdControlChan,
&RSZ_mdCreateChan,
&RSZ_mdDeleteChan,
};
```

The name of function table will be RSZMD_FXNS.
The function table type will be IOM_Fxns.

Driver open and close

```
/* open a logical channel */.
GIO_Handle      rszHandle;
Int SegId =0; /* for external memory */
rszHandle = GIO_create("/resizer",IOM_INOUT,NULL,(void *)&segId,&gioAttrs);
if(rszHandle == NULL) {
        printf(“open resizer channel failed.\n”)
        exit(-1);
}

/* close the logic channel */
GIO_delete(rszHandle);
```

Setup resizing parameters

```
PSP_RSZparams params;

/* setup the parameter here */
params.in_hsize = 720;
params in_vsize = 240;  /* only 1 filed of NTSC image */
params.in_pitch = 720*2
params.cbilin = _RSZ_CBILIN_DISABLE;      /* filter with luma for low pass */
```

```
params.pix_fmt = PSP_RSZ_PIX_FMT_UYVY;
params.out_hsize = 360;
params.out_vsize = 120;
params.out_pitch = 360*2;
params.hfilt_coeffs = hcoeffs;
params.vfilt_coeffs = vcoeffs;

/* disable the luminance enhancer */
params.yenh_params.type = PSP_RSZ_YENH_DISABLE;

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

Perform the resizing operation

```
PSP_RSZresize resize;

resize.in_buf = inbufs[0]; /* malloced buffer or a user pointer */
resize.out_buf = outbufs[0];

/* perform the resizing operation */
GIO_control(rszHandle, PSP_RSZ_IOCTL_RESIZE, &resize);
```