TII DM6437 VPSS Drivers

Histogram API Specifications

Release Version 1.10.00
January 14, 2008

Table of Contents

Article I. Revision History

Date	Version	Changes	Author
October 9, 2006	Draft 0.01	Created	EI5
October 10, 2006	Issue 1.00	Updated as per technical review comments	EI5
November 19, 2006	Issue 1.01	1) PSP_HISTparam structure changed 2) Name of enqueue/dequeue command changed 3) Buffer structure, will be passed with, enqueue/dequeue command is changed. 4) Module name & file names are changed. 5) Description of DDC & LLC layer is modified. 6) Type of status parameter & Description of mode and optargs is changed in GIO_CREATE. 7) Description of appCallback is changed in GIO_SUBMIT. 8) Type of args parameter is changed in GIO_CONTROL. 9) Values of PSP_HIST_IP_SOURCE_CCDC & PSP_CFA_BAYER_PATTERN are changed. 10) Description is changed in PSP_VPSS_DEQUEUE command. 11) Function table name is changed & initialization function name is removed in section 5.1. 12) Sections 5.2, 5.3, 5.4 & 5.5 are changed.	EI5
November 20, 2006	Presilicon Release 0.3.0	Release to TI	EI5
November 29, 2006	Post-silicon Release 0.3.0	Release to TI	EI5
December 4, 2006	Post-silicon Release 0.3.0GA Patch Release 1.00.03.01	1) Two parameters cfaPattern & ipSource are removed from PSP_HISTparam structure	EI5
December 26, 2006	Post-silicon Release 0.3.0	1) Modified after removing PSP_HISTedmaobj structure.	EI5
December 30, 2006	Post-silicon Release 0.4.0	Release to TI	EI5
June 22, 2007	1.00.01	GA Patch Release 1	Anuj Aggarwal
June29,2007	1.00.02	Modified Release Version	Amit Chatterjee

July 18, 2007	1.00.03	Modified Release Version	Maulik Desai
November 29, 2007	1.00.04	PSP merge package changes –modified release version and directory structure changes	Sivaraj R
January 14, 2008	1.00.05	PSP_HIST_IOCTL_SET_SEM_TIMEOUT IOCTL added	Sivaraj R

1. Overview

1.1 Purpose and Scope

This document provides APIs for the proposed driver for the Histogram 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.

1.2 Names and Terminology

The module name of the Histogram driver shall be “histogram". Hence the name of the top level files which will directly interact with application shall be “dda_histIOM.c” and “dda_histIOM.h”. These above files will interact with the “dda_hist.c” and “dda_hist.h”. Thereafter the “dda_hist.c” will interact with the “ddc_hist.c” and “ddc_hist.h”. Finally, the files related to hardware block will be referred as the “llc_hist.c” and “llc_hist.h”.

1.3 Architecture

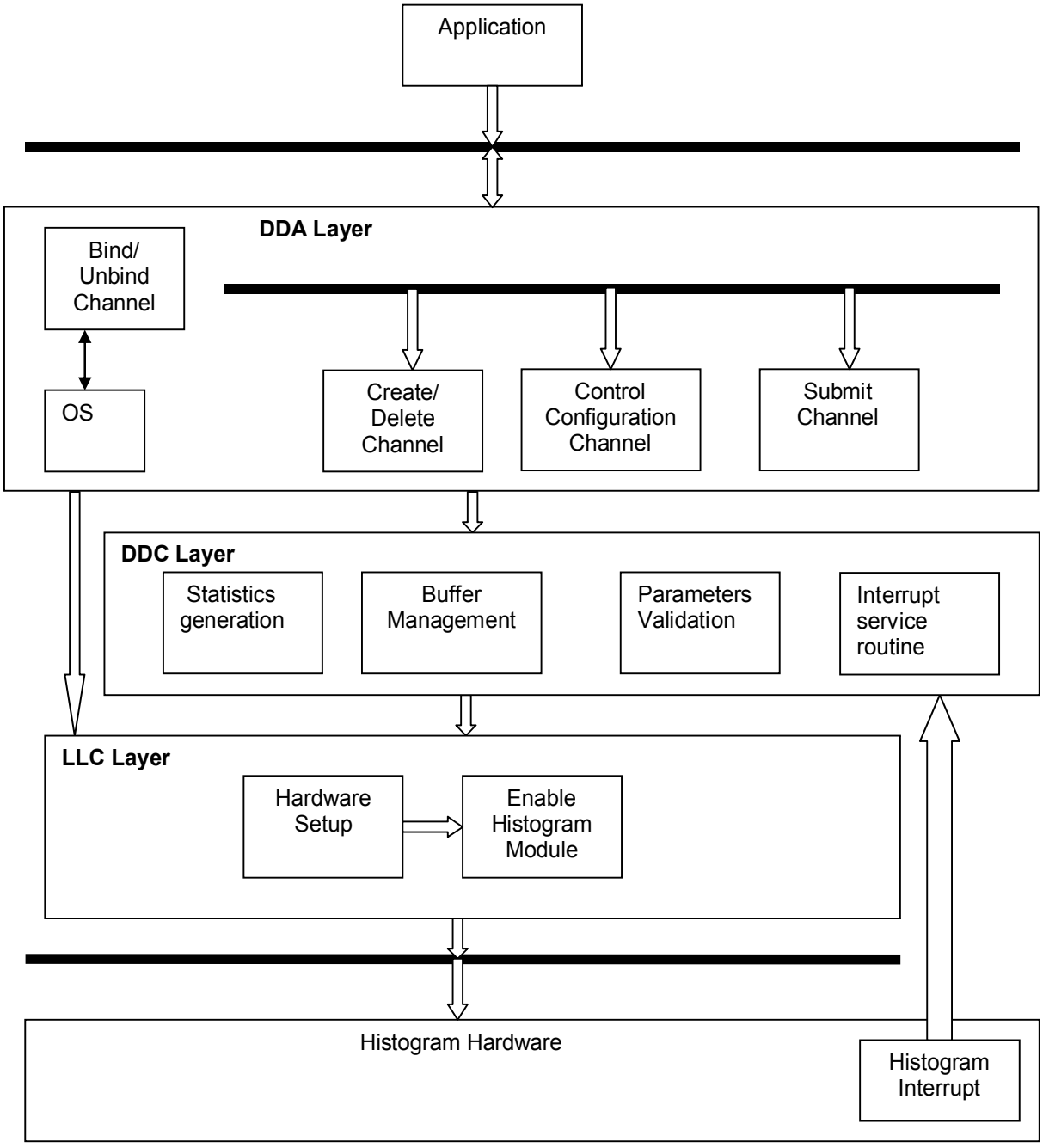

Figure 1. Top-Level Block Diagram of Histogram

The Histogram 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
- Create/delete
- Various controls to configure HW
- Reading of statistics

DDC Layer
This layer is mainly responsible for buffer management and providing the statistics to upper layer. It also validates the parameters sent by the application. It also handles the ISR.

LLC Layer
This layer is responsible for the actual configuration of the Histogram hardware by writing to the Histogram MMRs. It is pretty much OS agnostic and hardware centric. It configures & enables/disables the Histogram module.

1.4 Critical Features and Implementation

None

2. Application Level APIs

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

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

2.1 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 "/histogram"

Mode

The mode argument specifies the mode in which the device is to be opened. Histogram driver will only support IOM_INPUT mode.

Status

If the status parameter is non-NULL, a status value is placed at the address specified by the status parameter.

optArgs

This parameter is optional or can be used in driver specific manner. For histogram driver it is a pointer to EDMA device handle. Application must pass the EDMA device handle.

Attrs

The attrs parameter is a pointer to a structure of type GIO_Attrs.

Description
Open a logical channel. Multiple channel open will not be supported by the Histogram driver.

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.

2.2 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 occurs

2.3 GIO_CONTROL

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

GioChan
Handle to an instance of the device
cmd
Control functionality command to perform
args
Pointer to 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 Histogram control requests are located in the “psp_histogram.h” header file.

Return Value
IOM_COMPLETED on success and negative value if error.

2.4 GIO_SUBMIT

Synopsis
int GIO_submit(GIO_Handle gioChan, Uns cmd, Ptr bufp, Uns *pSize, GIO_AppCallback *appCallback);

GioChan
Handle to an instance of the device
cmd
Specified mini-driver command
bufp
Pointer to data structure for buffer data
pSize
The pSize parameter points to the size of the buffer structure.
appCallback
Pointer to callback structure. It will be NULL for Histogram driver.

Description
An application calls GIO_submit to submit the buffer to driver and to read the statistics from the driver.

Return Value
IOM_COMPLETED on success and negative value if error.

3. Histogram module Controls

3.1 PSP_HIST_IOCTL_SET_PARAMS

Name
PSP_HIST_IOCTL_SET_PARAM – sets the Histogram hardware parameters associated with this channel

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

Arguments
gioChan
Handle to an instance of the device
cmd
PSP_HIST_IOCTL_SET_PARAMS
argp
pointer to the PSP_HISTparam structure

Description
This ioctl is used to set the parameters of the Histogram hardware. Description about the parameters and possible values of few of the parameter is given below.

```
typedef struct _PSP_HISTparam{
   /**< clearing of Histogram output data after read          */
   PSP_HISTclearopdata        clearOpDataAfterRead;
   /**< bins per Histogram                                    */
   PSP_HISTbins               bins;
   /**< right shift of data for Histogram binning             */
   PSP_HISTdatarightshift     shift;

   /**< white balance gain for all colors                     */
   Uint8                      wbGain[4];

   /**< Region boundaries for all regions                     */
   PSP_HISTregionboundries    region[4];

} PSP_HISTparam;

typedef enum _PSP_HISTclearopdata{
   /**< Do not clear Histogram output data after read        */
   PSP_HIST_NO_CLEAR_AFTER_READ = 0,
   /**< Clear Histogram output data after read               */
   PSP_HIST_CLEAR_AFTER_READ

} PSP_HISTclearopdata;

typedef enum _PSP_HISTbins{
   /**< 32 bins per Histogram                          */
   PSP_HIST_32BINS  = 0,
   /**< 64 bins per Histogram                          */
   PSP_HIST_64BINS,
   /**< 128 bins per Histogram                         */
```

```
    PSP_HIST_128BINS,
    /**< 256 bins per Histogram                    */
    PSP_HIST_256BINS

} PSP_HISTbins;

typedef enum _PSP_HISTdatarightshift
{
    /**< Right shift data by 0 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_ZERO  = 0,
    /**< Right shift data by 1 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_ONE,
    /**< Right shift data by 2 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_TWO,
    /**< Right shift data by 3 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_THREE,
    /**< Right shift data by 4 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_FOUR,
    /**< Right shift data by 5 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_FIVE,
    /**< Right shift data by 6 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_SIX,
    /**< Right shift data by 7 for Histogram bining          */
    PSP_HIST_DATA_RSHIFT_SEVEN

} PSP_HISTdatarightshift;

typedef struct PSP_HISTregionboundries{
    /**< horizontal start position                 */
    Uint16     hStart;
    /**< horizontal end position                   */
    Uint16     hEnd;
    /**< vertical start position                   */
    Uint16     vStart;
    /**< vertical end position                     */
    Uint16     vEnd;
} PSP_HISTregionboundries;
```

Return Value
IOM_COMPLETED on success and negative value if error.

3.2 PSP_HIST_IOCTL_GET_PARAM

Name
PSP_HIST_IOCTL_GET_PARAM – get the Histogram hardware parameters associated with this logic channel.

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

Arguments
gioChan
Handle to an instance of the device
cmd
PSP_HIST_IOCTL_GET_PARAMS

argp

pointer to the PSP_HISTparam structure

Description

This ioctl is used to get the Histogram hardware settings.

Return Value

IOM_COMPLETED on success and negative value if error.

3.3 PSP_HIST_IOCTL_SET_SEM_TIMEOUT

Name

PSP_HIST_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_HIST_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.

4. Histogram Submit Commands

4.1 PSP_VPSS_QUEUE

Name
PSP_VPSS_QUEUE – provides the buffer to the drive to capture the data.

Synopsis

GIO_submit(GIO_Handle gioChan, Uns cmd, struct PSP_HISTbuffer* bufp, Uns *pSize, NULL);

GioChan
Handle to an instance of the device
cmd
PSP_VPSS_QUEUE
bufp
Pointer to PSP_HISTbuffer structure
pSize
The pSize parameter points to the size of the buffer structure.
appCallback
NULL

Description
It provides the buffer and buffer size to the driver to store the data.

```
typedef struct _PSP_HISTbuffer{
   /**< Head node to queue list                    */
   PAL_OsListNodeHeader     hNode;
   /**< buffer address                             */
   Ptr                      ramIpAddr;
   /**< buffer size                                */
   Uint32                   bufferSize;
   /**< time of captured data                      */
   Uint32                   timeStamp;

} PSP_HISTbuffer;
```

Return Value
IOM_COMPLETED on success and negative value if error.

4.2 PSP_VPSS_DEQUEUE

Name
PSP_VPSS_DEQUEUE – provides the captured data to the application.

Synopsis

GIO_submit(GIO_Handle gioChan, Uns cmd, struct PSP_HISTbuffer* bufp, Uns *pSize, NULL);

GioChan

Handle to an instance of the device

cmd

PSP_VPSS_DEQUEUE

bufp

Pointer to PSP_HISTbuffer

pSize

The pSize parameter points to the size of the buffer structure.

appCallback

NULL

Description

It provides the captured data to the application.

Return Value

IOM_COMPLETED on success and negative value if error.

5. Usage Examples

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

5.1 Registration of Histogram 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 “histogram”.

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

4. In the Properties dialog, specify the name of the function table as”_HISTMD_FXNS” and function table type as “IOM_Fxns”.

5.2 Driver open and close

```
/* open a logical channel */.
GIO_Handle      histHandle;
EDMA3_DVR_Handle hEdma;
Int gioStatus;

/* Application must pass the EDMA device handle */

histHandle = GIO_create("/histogram", IOM_INPUT, &gioStatus, hEdma, &gioAttrs);
if(NULL == histHandle) {
        printf(“\n Histogram channel open failed. \n”)
        exit(-1);
}

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

5.3 Setup Histogram parameters

```
PSP_HISTparam params;
/* set the structure parameter as per requirement */
PSP_HISTparam getParams;

/* configure the logic channel */
GIO_control(histHandle, PSP_HIST_IOCTL_SET_PARAM, &params);

/* get the parameter of the logic channel */
GIO_control(histHandle, PSP_HIST_IOCTL_GET_PARAM, &getParams);

/* verify the parameters */
```

5.4 Performing the enqueue operation

```
PSP_HISTbuffer          buffer;
Int size;
```

```
buffer.ramIpAddr = MEM_alloc(ddr2_segmentID, 1024*4, 128);
memset(buffer.ramIpAddr, 0, 1024*4);
buffer.bufferSize = 1024;
size = sizeof(PSP_HISTbuffer);

/* submit the empty buffer */
GIO_submit(histHandle, PSP_VPSS_QUEUE, (Ptr)(&buffer), (Ptr)(&size), NULL);
```

5.5 Performing the dequeue operation

```
PSP_HISTbuffer          *outbuffer;
int size;

size = sizeof(PSP_HISTbuffer);

/* get the buffer which contains statistics */
GIO_submit(histHandle, PSP_VPSS_DEQUEUE, (Ptr)(&outbuffer), (Ptr)(&size), NULL);
```