Texas Instruments

# DSP/BIOS Histogram Device Driver

# *User's Manual*

***May 2008*** ***Platform Software Group***

***Release 1.10.00***

## 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 Histogram 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 |
| PALSYS | Platform Abstraction Layer System |
| EDMA3 | Third Party Direct Memory Access Controller |

## Revision History

| Date | Author | Comments | Version |
|---|---|---|---|
| November 18,2006 | EI5 | Created the document | 1.0.0 |
| November 20,2006 | EI5 | Release to TII | Pre-silicon Release 0.3.0 |
| November 29,2006 | EI5 | Release to TII | Post-silicon Release 0.3.0 |
| December 26,2006 | EI5 | 1) Modified after removing PSP_HISTedmaobj structure. | Post-silicon Release 0.3.0 |
| January 2,2007 | EI5 | 1) Modified for change of buffer management design.<br>2) Constraints section added. | Post-silicon Release 0.4.0 |
| January 16,2007 | EI5 | 1) Bios version has been changed. | Post-silicon Release 0.4.2 |
| January 25,2007 | EI5 | 1) PALOS, CSL, EDMA3 & PALSYS added in Abbreviations section.<br>2) CCS version changed. | Post-silicon Release 0.5.0 |
| March 5, 2007 | EI5 | 1) Description for PSP_VIDEO_PATH_ENABLE is added in section 4.1.2.<br>2) Region Overlapping added in constraints section.<br>3)Section 4.1 is modified | Post-silicon Release 0.6.0 |
| April 21, 2007 | EI5 | Release to TII | Post-silicon Release 0.7.0 |
| 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 | Maulik Desai | 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_HIST_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 Histogram Device Driver for DM6437 SOC.

## 1.1 H/W S/W Support

This Histogram 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:

**HISTOGRAM IOM –** Application facing, OS Specific Adaptation of Histogram Device Driver

**HISTOGRAM DDC –** OS Independent part of Histogram Driver Core

**HISTOGRAM CSL –** The low-level Histogram h/w abstraction module

**System components:**

**PALOS –** DSP/BIOS Abstraction
**PALSYS -** DSP/BIOS Abstraction
**EDMA3 –** Driver to utilize the DSP/BIOS data transfer service

**CSL –** Non functional h/w abstraction.

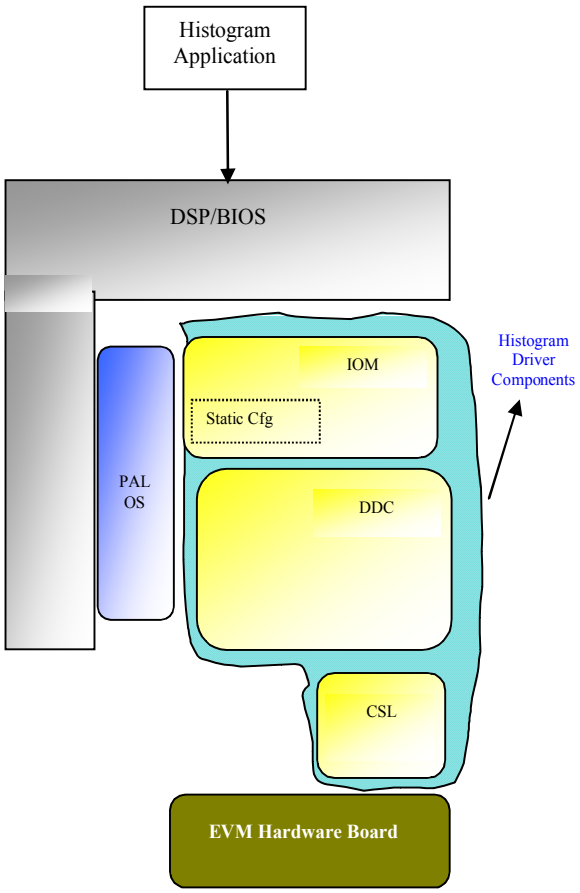


***Figure 1 Device Driver Functional Decomposition***

## 1.3 Default Driver Configuration

Histogram 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 parameter it will return the corresponding error code.

## 1.4 Driver Capabilities

The significant driver features are:

- Histogram driver supports input from CCDC.
- Histogram driver supports output to DDR.
- Histogram driver supports configuration of White Balance Gain.
- Histogram driver supports Binning for multiple regions.
- Histogram driver works synchronously.
- Histogram driver provides time stamp to the captured statistics.
- Histogram driver manage the buffers using Queue/Dequeue mechanism. It will automatically enable HW engine when first buffer is queued.

## 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:

**Input Pattern Type**

- Histogram driver will provide support for only Bayer Pattern Input.

**Input Type**

- Histogram driver will support only CCDC Input.

**Buffer management**

- Application can deque one buffer less than number of buffers it has enqueued. I.e. If N buffers are enqueued, then N-1 buffers can be dequeued.

**Region Overlapping**

- In case of overlapping regions the statistics would be accumulated in only 1 region which is of higher priority.

**General**

- Histogram engine must be enabled prior to CCDC engine.
- Histogram interrupts are dependent on CCDC interrupts. In worst case if the ccdc interrupts comes late then histogram interrupts will also come late. So it might be possible to change the semaphore (SEMISR) time out value to work histogram driver successfully.

# Chapter 2

# INSTALLATION GUIDE

## 2.1 Component Folder

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

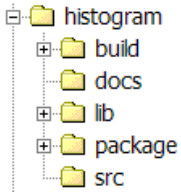


***Figure 2 Histogram Driver Directory Structure***

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

- **build:** This folder contains histogram driver library project file. The generated driver library shall be included in the application where HISTOGRAM 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 histogram 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 is 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 "src" folder and the driver library is provided in "\drivers\histogram\lib" 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 and release 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.

- **HIST_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 HISTOGRAM

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,**<br>**Int mode,**<br>**Int* status,**<br>**Ptr chanParams,**<br>**GIO_Attrs* attrs)** |

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

Note:

- It is assumed that the application has initialized the memory pool from which it desires the allocation to happen.
- Application must pass the handle to EDMA3 device.

**Parameters:**

| | |
|---|---|
| *name* | [IN] Name of the device to open |
| *mode* | [IN] Mode in which device is to be opened |
| *dtatus* | [OUT] Address to which driver returns status |
| *chanParams* | [IN] EDMA3 device handle |
| *attrs* | [IN] pointer to GIO_attrs structure |

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

**Example**:

```
/* Histogram device handle */
GIO_Handle histHandle;
```

```
/* Histogram parameter configuration structure. */
PSP_HISTparam params;
PSP_HISTparam outparams;

/* Histogram buffer structure to get the statistics */
PSP_HISTbuffer    buffer[2];
Int sizeOfbuff;

/* pointer to buffer which contains the statistics */
PSP_HISTbuffer *outbuff = NULL;

Int gioStatus;

/* Application must open the EDMA3 device channel prior to histogram
channel is created. */

/* Initialize the CCDC device for 10bit raw mode capture. */

histHandle = GIO_create(deviceName, IOM_INPUT, hEdma, NULL);

deviceName is passed as "/histogram". Instance is declared in the
configuration file.
```

### 3.1.2 GIO_delete

| **Int GIO_delete** | **(GIO_Handle gioChan)** |
|---|---|
| | |

This function deletes the driver channel.

**Parameters:**

| *gioChan* | [IN] GIO Handle for the channel |
|---|---|
| | |

**Returns:**
IOM_COMPLETED

**Example**:

```
/* delete the instance of histogram channel */
GIO_delete(histHandle);

/* delete the instance of CCDC device */

/* delete the instance of EDMA3 device */
```

### 3.1.3 GIO_control

| | |
|---|---|
| **PSP_Result GIO_control** | **(GIO_Handle *gioChan*,<br>Int *cmd*,<br>Ptr *cmdArg*)** |

This function handles the IOCTLs for the histogram 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 – if the argument passed is not appropriate
IOM_EBADMODE – if the device state is not proper

**Example**:

```
/* clear histogram output memory contents after read. */
params. clearOpDataAfterRead = PSP_HIST_CLEAR_AFTER_READ;

/* number of bins per histogram */
params.bins = PSP_HIST_256BINS;

/* shift amount to input data of histogram */
params.shift = PSP_HIST_DATA_RSHIFT_TWO;

/* white balance for all colors. */
params. wbGain[0] = 0x20;      /* 1 */
params. wbGain[1] = 0x20;      /* 1 */
params. wbGain[2] = 0x20;      /* 1 */
params. wbGain[3] = 0x20;      /* 1 */

/* region 0 boundaries */
params. region[0].hStart = 0;
params. region[0].hEnd = 320;
params. region[0].vStart = 0;
params. region[0].vEnd = 240;

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

/* get the configured parameter */
GIO_control(histHandle, PSP_HIST_IOCTL_GET_PARAM, &outparams);
```

### 3.1.4 GIO_submit

| | |
|---|---|
| **PSP_Result GIO_submit** | **(GIO_Handle gioChan,** |

| | | |
|---|---|---|
| | **Uns** | **cmd,** |
| | **Ptr** | **bufp,** |
| | **Uns** | ***pSize,** |
| | **GIO_AppCallback** | ***appCallback)** |

This function does the enque/deque operation for the histogram driver. The histogram module will automatically be enabled when first buffer is enqued & will automatically be disabled when enqueue stack is empty.

Note:

- CCDC must be enabled only after the histogram module is enabled.
- Video Port must be enabled.
- Input buffer address must be 128 bytes aligned & it must be cache invalidated before use once statistics are stored in it.

**Parameters:**

| | |
|---|---|
| *gioChan* | [IN] GIO Handle for the channel |
| *cmd* | [IN] Specified mini-driver command |
| *bufp* | [IN/OUT]Pointer to data structure of buffer data |
| *pSize* | [IN]Size of data buffer |
| *appCallback* | [IN]Pointer to call back function |

**Returns:**
IOM_COMPLETED if successful or else suitable error code is given.

**Example**:

```
sizeOfbuff = sizeof(PSP_HISTbuffer);

/* Buffer must be 128 bytes aligned */
buffer[0].ramIpAddr = MEM_alloc(ddr2_segmentID, 1024 * sizof(Int),
128);
buffer[1].ramIpAddr = MEM_alloc(ddr2_segmentID, 1024 * sizof(Int),
128);
buffer[0].bufferSize = 1024;  /* In terms of word(4 bytes) */
buffer[1].bufferSize = 1024;  /* In terms of word(4 bytes) */

/* submit the first empty buffer to enable the histogram */
GIO_submit(histHandle, PSP_VPSS_QUEUE, &buffer[0], (Ptr)(&sizeOfbuff),
NULL);

/* enable CCDC */

/* submit the second empty buffer to get the statistics */
GIO_submit(histHandle, PSP_VPSS_QUEUE, &buffer[1], (Ptr)(&sizeOfbuff),
NULL);

/* buffer pointed by outbuff will contain the captured statistics */
GIO_submit(histHandle, PSP_VPSS_DEQUEUE, &outbuff, (Ptr)(&sizeOfbuff),
NULL);

/* Buffer containing the statistics must be cache invalidated before
the statistics are dump into the file. */
```

```
PAL_osCacheInvalidate(
      (PAL_osCacheMemAddrSpace)NULL,
      (Uint32)(outbuff->ramIpAddr),
      outbuff->bufferSize * sizeof(Int));

/* Dump the statistics into the file. */
```

## 3.2 Data Structures Configuration defines

The file **psp_histogram.h** has the **_PSP_HISTregionboundries** data structure that is passed for specifying the region boundaries. The structure is passed in GIO_control .The parameters of the structure are explained below:

**1) Region Boundaries Data Structure**

| Parameter | Description |
|---|---|
| hStart | Horizontal start position for region. |
| hEnd | Horizontal end position for region.<br>If the "end" is programmed as the same value of "start", then the region size is treated as 0. In all cases, the total number of pixels processed is "End - Start + 1". Therefore, the minimum region dimension is at least 2. |
| vStart | Vertical start position for region. |
| vEnd | Vertical end position for region.<br>If the "end" is programmed as the same value of "start", then the region size is treated as 0. In all cases, the total number of pixels processed is "End - Start + 1". Therefore, the minimum region dimension is at least 2. |

**NOTE:** These values can be any digit but they should not exceed the CCDC frame size.

The file **psp_histogram.h** has the **_PSP_HISTparams** data structure that is passed for configuration of histogram parameters in the GIO_control. The parameters of the structure are explained below:

**2) Histogram Configuration parameters Data Structure**

| Parameter | Description |
|---|---|
| clearOpDataAfterRead | Clearing of Histogram output data after read.<br>Any value from PSP_HISTclearopdata enumeration can be assigned to this variable. |
| Bins | Bins per Histogram.<br>Any value from PSP_HISTbins enumeration can be assigned to this variable. |
| Shift | Right shift of data for Histogram binning.<br>Any value form PSP_HISTdatarightshift enumeration can be assigned |

| | |
|---|---|
| | to this variable. |
| wbGain[4] | White balance gain for all colors.<br>It is set within the range of x0~x7.96875(7+31/32). The decimal point is located between WG01[5] and WG01[4]. |
| region[4] | Region boundaries for all regions. |

The file **psp_histogram.h** has the **_PSP_HISTbuffer** data structure that is passed for configuration of parameters required during enqueue and dequeue operation. The structure is passed via GIO_submit. The parameters of the structure are explained below:

**3) Enqueue/Dequeue Data Structure**

| Parameter | Description |
|---|---|
| hNode | Head node for the entry of the buffer in queue list. |
| ramIpAddr | Input Buffer base address to store data. It must be 128 byte aligned. |
| timestamp | Time stamp (Milliseconds) to the captured statistics from histogram module. |
| Size | Size of the input buffer.<br>It should be 512 words (4 bytes) if 32bins per histogram is configured else it should be 1024. Buffer size less then this will provide insufficient statistics. |

The file **psp_histogram.h** has the **_PSP_HISTedmaobj** data structure that is passed to GIO_control to provide the edma information to hardware. The parameters of the structure are explained below:

## 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_HIST_REGION_MAX_HPOS | Maximum horizontal position of the histogram region. Its value is 16384. |
| Macro | PSP_HIST_REGION_MAX_VPOS | Maximum vertical position of the histogram region. Its value is 16384. |
| Macro | PSP_HIST_NUM_INSTANCES | Indicates that maximum number of histogram instance supported by driver. Its value is 1. |
| Enumeration | PSP_HIST_NO_CLEAR_AFTER_READ | Indicates that Histogram output memory content |

| | | |
|---|---|---|
| | | should not be cleared after the data is read. Its value is 0. |
| Enumeration | PSP_HIST_CLEAR_AFTER_READ | Indicates that Histogram output memory content should be cleared after the data is read. Its value is 1. |
| Enumeration | PSP_HIST_32BINS | Indicates that 32 bins per histogram are required. Its value is 0. |
| Enumeration | PSP_HIST_64BINS | Indicates that 64 bins per histogram are required. Its value is 1. |
| Enumeration | PSP_HIST_128BINS | Indicates that 128 bins per histogram are required. Its value is 2. |
| Enumeration | PSP_HIST_256BINS | Indicates that 256 bins per histogram are required. Its value is 3. |
| Enumeration | PSP_HIST_DATA_RSHIFT_ZERO | Indicates that input data to histogram module is shifted by zero. Its value is 0. |
| Enumeration | PSP_HIST_DATA_RSHIFT_ONE | Indicates that input data to histogram module is shifted by one. Its value is 1. |
| Enumeration | PSP_HIST_DATA_RSHIFT_TWO | Indicates that input data to histogram module is shifted by two. Its value is 2. |
| Enumeration | PSP_HIST_DATA_RSHIFT_THREE | Indicates that input data to histogram module is shifted by three. Its value is 3. |
| Enumeration | PSP_HIST_DATA_RSHIFT_FOUR | Indicates that input data to histogram module is shifted by four. Its value is 4. |
| Enumeration | PSP_HIST_DATA_RSHIFT_FIVE | Indicates that input data to histogram module is shifted by five. Its value is 5. |

| | | |
|---|---|---|
| Enumeration | PSP_HIST_DATA_RSHIFT_SIX | Indicates that input data to histogram module is shifted by six. Its value is 6. |
| Enumeration | PSP_HIST_DATA_RSHIFT_SEVEN | Indicates that input data to histogram module is shifted by seven. Its value is 7. |
| Enumeration | PSP_HIST_IOCTL_SET_PARAMS | Control Command to set the histogram parameters. _PSP_HISTparams must be passed with valid configuration values to configure the histogram. |
| Enumeration | PSP_HIST_IOCTL_GET_PARAMS | Control Command to get the histogram parameters. Returned parameters will be available in _PSP_HISTparams type of structure. |
| Enumeration | PSP_HIST_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_VPSS_DEQUEUE | This command gets the buffer containing the statistics form histogram. Returned buffer will be of type _PSP_HISTbuffer. |
| Enumeration | PSP_VPSS_QUEUE | This command submits the empty buffer to histogram. _PSP_HISTbuffer must be passed with this command. |

# CHAPTER 4

# EXAMPLE APPLICATION

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

## 4.1 Writing Applications for Histogram

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

### 4.1.1 File Inclusion

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

1. **psp_histogram.h**

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

### 4.1.2 Driver Initialization

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

To have Histogram device driver included in the application, corresponding TCI file have to be included in BIOS TCF i.e. "*dm6437_histogram.tci*" must be included in BIOS TCF file of the application. This file can be found in video sample directory.

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

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

### 4.1.3 Dependent Projects/Libraries

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

- Histogram
- VPFE
- I2C
- Previewer
- EDMA3 DRV
- EDMA3 RM
- EDMA3 DRV Sample
- 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 Histogram Sample Application

### 4.2.1 Introduction

This psp_bios_hist_st_sample.pjt application demonstrates histogram for 256bins configuration. Before running the sample application of Histogram run the previewer on the fly application to ensure that sensor is configured correctly and input image is stable.

### 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\histogram* folder. The sample can be rebuilt directly from this project file using Code Composer studio.
Before building sample application make sure that the VPFE library is built with PSP_VIDEO_PATH_ENABLE Macro enabled in psp_vpfe.h.

### 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 studio.
After running the sample application of Histogram, execute the previewer on the fly application to ensure output image is still stable.

# CHAPTER 5

# HISTOGRAM PERFORMANCE RESULT

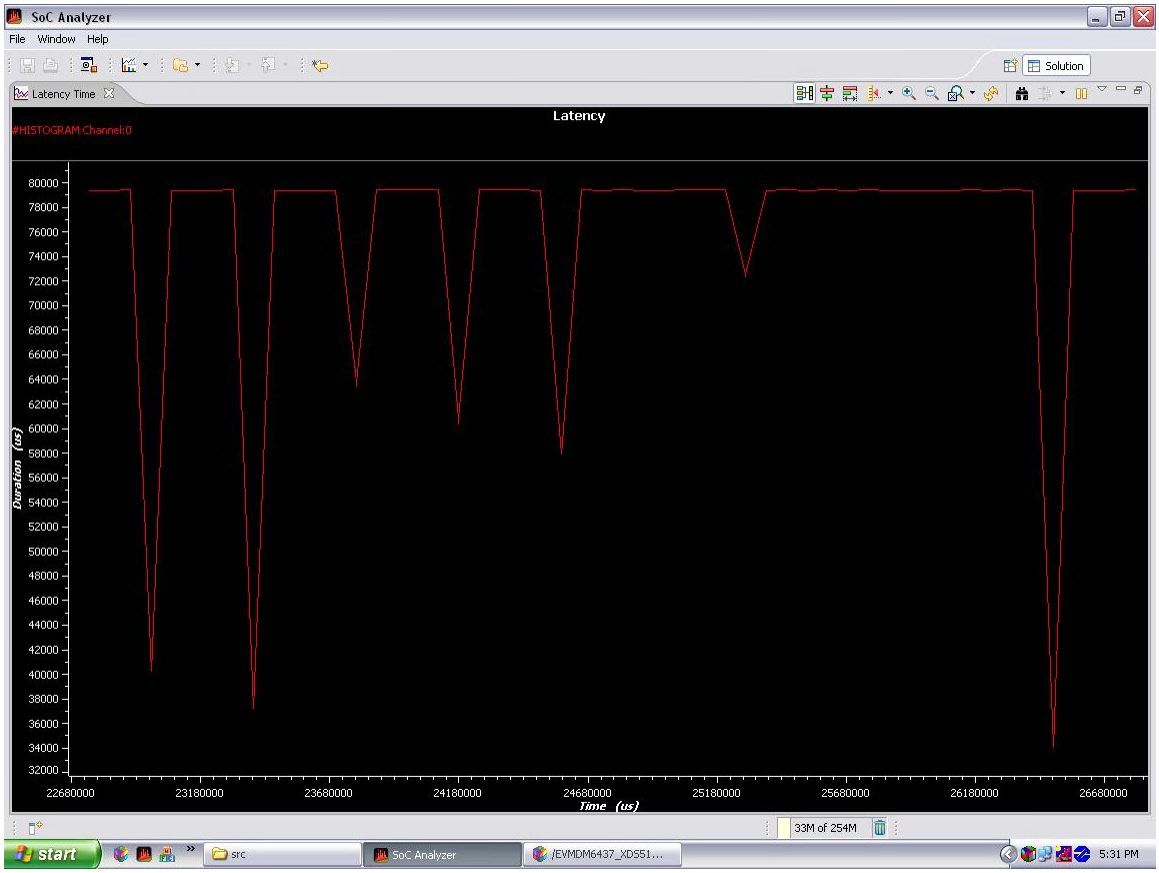


***Figure 3 Latency graph of Histogram***

**Note:** The Driver Performance Characteristics can be included once testing is done on DM6437 SOC. The graph is taken for Histogram operating in edma mode.