Texas Instruments

DSP/BIOS H3A 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

Preface

Read This First

About This Manual

The API reference guide serves as a software programmer's handbook for working with the H3A 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
CSL	Chip Support Library
PALOS	Platform Abstraction Layer Operating System

Revision History

Date	Author	Comments	Version
November 15,2006	EI4	Created the document	1.0.0
November 20,2006	EI4	Release to TII	Pre-silicon Release 0.3.0
November 29,2006	EI4	Release to TII	Post-silicon Release 0.3.0
December 20,2006	EI4	Description added for APIs and data structures	Post-silicon Release 0.3.0
January 2, 2007	EI4	Section 1.6 added.	Post-silicon Release 0.4.0
January 16, 2007	EI4	Bios version changed.	Post-silicon Release 0.4.2
January 25, 2007	EI4	Abbreviations for CSL, PAL added. IOCTL command reference provided in GIO_control. CCS version changed.	Post-silicon Release 0.5.0
March 06, 2007	EI4	In section 1.6 constraints are modified. In section 4.1 and 4.2 is modified.	Post-silicon Release 0.6.0
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_H3A_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 H3A Device Driver for DM6437 SOC.

1.1 H/W S/W Support

This H3A Device driver has been developed for the DSP/BIOS operating system using 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:

H3A IOM – Application facing, OS Specific Adaptation of H3A Device Driver

H3A DDC –OS Independent part of H3A Driver Core

H3A CSL –The low-level H3A h/w abstraction module

System components:

PALOS – DSP/BIOS Abstraction

CSL – Non functional h/w abstraction.

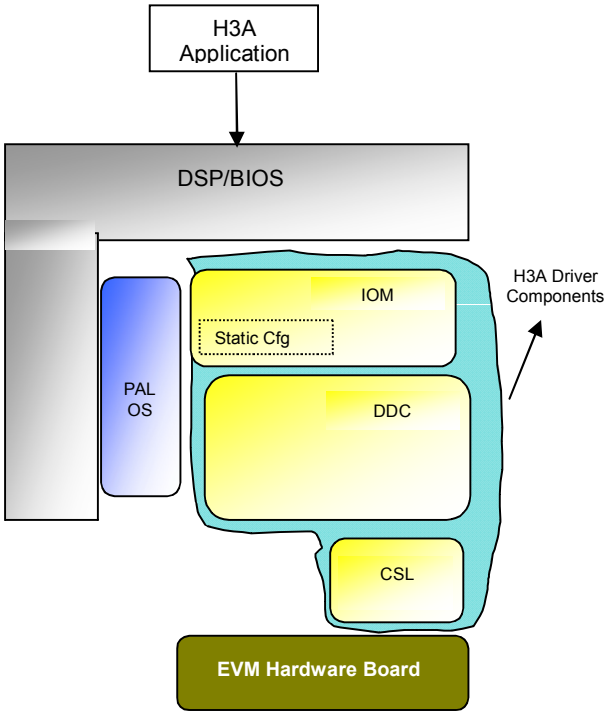

Figure 1 Device Driver Functional Decomposition

1.3 Default Driver Configuration

H3A 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 parameter at the time of channel creation, the handle for the corresponding channel shall not be created and driver will return NULL.

1.4 Driver Capabilities

The significant driver features are:

- AF and AEW driver supports input from CCDC
- AF and AEW driver supports output to DDR.
- AF and AEW driver supports single application.
- AF and AEW driver works synchronously.
- AF driver supports user configurable IIR Filter coefficients.
- AF driver supports user configurable paxels.
- AF driver supports RGB positioning.
- AF driver supports configuration of A-Law and Horizontal Median Filter.
- AEW driver supports user configurable windows.
- AEW driver supports configuration of A-Law.
- AEW driver supports configuration of saturation check to set clipping value.
- AEW driver supports user configurable black windows.

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:

Raw Pattern Type

- H3A will provide support for only Bayer Pattern Input.

AEW Output Address

- AEW Output address must be on 64 byte boundaries.

AEW Window Configuration

- The width and height of the windows must be an even number.
- Sub-sampling windows can only start on even numbers.
- The minimum width of the AE/AWB windows must be 6 pixels.
- Support for up to 36 windows in horizontal direction
- Support for up to 128 windows in vertical direction
- Windows cannot overlap
- All windows must be accommodated in the image.

(Windows Horizontal Start + (Width of window * Windows Horizontal Count)) should be less then or equal to Image width

(Windows Vertical Start + (Height of window * Windows Vertical Count)) should be less then or equal to Image Height

AF Horizontal Median Filter

- If the median filter is enabled then the first two and last two pixels in the frame are not in the valid region. Therefore the Paxel start/end and IIR filter start positions should not be set within the first two and last two pixels.

AF Paxel Configuration

- The minimum width of the Paxel must be 6 pixels. The width and height of the Paxels must be an even number.
- Paxels cannot overlap and must be adjacent to one another.
- Paxel horizontal start value must be greater than or equal to the (IIR horizontal start position + 2)
- Support for up to 36 paxels in horizontal direction
- Support for up to 128 paxels in vertical direction
- All paxels must be accommodated in the image.

(Paxel Horizontal Start + (Width of Paxel * Paxel Horizontal Count)) should be less then or equal to Image width

(Paxel Vertical Start + (Height of Paxel * Paxel Vertical Count)) should be less then or equal to Image Height

AF Window Size

- The minimum width of the window must be 6 pixels. The width and height of the window must be an even number.

AF Output Address

- AF Output address must be on 64 byte boundary.

AF IIR Filter

- Horizontal Start Position for IIR Filter must be even.

- H3A interrupts are dependent on CCDC interrupts. In worst case if the ccdc interrupts comes late then H3A interrupts will also come late. So it might be possible to change the semaphore (SEMISR) time out value to work H3A driver successfully.

Chapter 2

Installation Guide

2.1 Component Folder

Upon installing the H3A driver the following directory structure is found in the driver's directory.

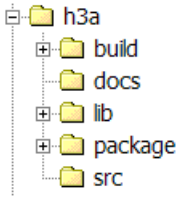

Figure 2 H3A Driver Directory Structure

This top level h3a folder contains h3a driver psp header file and XDC package files (package.bld, package.xdc and package.xs)

- **build:** This folder contains h3a driver library project file. The generated driver library shall be included in the application where H3A 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 h3a 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 ensure 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/h3a/build 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.
- **H3A_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 H3A

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

3.1 Functions for Auto Focus Driver

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 Auto Focus channel. GIO_create () populates static settings in driver object, formally creates/registers driver entry point with DSP/BIOS. Along with Auto Focus channel, CCDC Channel must be opened and configured.

Note:

- It is assumed that the application has initialized the memory pool from which it desires the allocation to happen.
- The AF should not be enabled for Foveon formatted input

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] Channel Type
attrs	[IN] pointer to GIO_attrs structure

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

Example:

```
void AF_open()
{
      GIO_Handle   afChan;
      PSP_H3AChannelType type;
      Int gioStatus;
```

```
      /* Configure CCDC Engine */
      /* open a logical channel */
      type= PSP_H3A_AF;
      afChan = GIO_create("/H3A",IOM_INOUT,&gioStatus,&type,&gioAttrs);
}
deviceName is passed as H3A. Instance is declared in the configuration
file. Type is type of the channel(AF or AEW).
```

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_EBADARGS – If the parameters passed are not correct or the channel has never been created

Example:

```
/* close the AF channel */
GIO_delete(afChan);
/* Close CCDC Channel */
GIO_delete(CcdcHandle);
```

Note:

- CCDC Handle must be deleted. If handle is not deleted, CCDC Engine will remain enabled. If the program is executed for the next time then Auto Focus may generate invalid statistics.

3.1.3 GIO_control

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

This function handles the IOCTLs for the Auto Focus 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 FIFO is not enabled or if the mode is not supported

Example:

```
/* No of PAXELS in vertical direction */
#define PAXEL_VT_COUNT      (8u)
/* No of PAXELS in the horizontal Direction */
#define PAXEL_HZ_COUNT      (8u)
/* Size of Buffer that will contain statistics */
#define H3A_AF_BUFFER_SIZE
      (PAXEL_VT_COUNT * PAXEL_HZ_COUNT *PSP_AF_PAXEL_SIZE)
static PSP_AFParams params =
{
    PSP_AF_ALAW_DISABLE,            /* A Law Status    */
    {
        PSP_AF_HMF_DISABLE,         /* HMF Status      */
        120u,                       /* Threshold value */
    },
    PSP_AF_GR_BG_BAYER,             /* RGB position    */
    /* IIR Filter Configuration */
    {
        2u,                         /* IIR Horizontal start */
        /*IIR Filter coefficients for set 0 */
        {
            64,
            0,
            0,
            21,
            22,
            21,
            0,
            0,
            -16,
            32,
            -16,
        },
        /*IIR Filter coefficients for set 1 */
        {
            64,
            0,
            0,
            21,
            22,
            21,
            0,
            -29,
            -16,
            32,
            -16,
        },
    },
```

```
    /* Paxel Configuration  */
    {
        8u,                          /* width    */
        8u,                          /* height   */
        8u,                          /* hzStart  */
        8u,                          /* vtStart  */
        PAXEL_HZ_COUNT,              /* hzCnt    */
        PAXEL_VT_COUNT,              /* vtCnt    */
        2u,                          /* lineIncr */
  },
    PSP_AF_ACCUMULATOR_SUMMED,       /* Accumulator Mode */
};

/* configure the logic channel */
GIO_control(afChan, PSP_H3A_IOCTL_SET_PARAM, &params);

Please refer section 3.6 for all IOCTL commands.
```

Note :
Some of the points to be taken into consideration while configuring CCDC are:

- Video Port should be enabled.
- AF does not support for the Foveon sensor. The input must be in bayer format.

Some of the points to be taken into consideration while configuring Auto Focus are:

- The paxel horizontal start value must be greater than or equal to the IIR horizontal start position.
- The width and height of the paxels must be an even number.
- The minimum width of the auto focus paxel must be 6 pixels.
- Paxels cannot overlap the last pixel in a line.
- Paxels must be adjacent to one another.

3.1.4 GIO_submit

PSP_Result GIO_submit	**(GIO_Handle** **Uns** **Ptr** **Uns** **GIO_AppCallback**	**gioChan,** **cmd,** **bufp,** ***pSize,** ***appCallback)**

This function handles the IOCTLs for the H3A driver.

Parameters:

gioChan	GIO Handle for the channel
cmd	Specified mini-driver command.
bufp	Pointer to data structure of buffer data
pSize	Size of data buffer structure.
appCallback	Pointer to call back function.

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

Example:

```
GIO_Handle afChan;
```

```
Int bufferSize;
PSP_H3ABufferbuffer;
PSP_H3ABuffer dequebuffer;

/* Size of the buffer Data Structure*/
bufferSize = sizeof(PSP_H3ABuffer);
/* For proper cache operation,
   Buffer size should be multiple of 128 bytes */
buffer.size = H3A_AF_BUFFER_SIZE;

/* Buffer must be 128 bytes aligned */
buffer.ramIpAddr  = (void *)MEM_alloc(0, buffer.size,(128u));

/* submit the empty buffer */
GIO_submit(afChan, PSP_VPSS_QUEUE, &buffer, &bufferSize, NULL);

/AF Engine is enabled */
/* Enable CCDC Engine */

GIO_submit(afChan, PSP_VPSS_DEQUEUE, &dequebuffer, &bufferSize, NULL);
/* Invalidate the cache */
```

Note :

- The input to the Auto Focus module is the video port interface of the CCD Controller, In order to guarantee that data from the CCD Controller is not missed, the Auto Focus should be enabled prior to the CCD Controller.
- Cache must be invalidated to ensure that data is read from SDRAM/DDRAM.
- For proper cache operation, buffer size should be multiple of 128 bytes (due to cache line size).

The behavior of submit call for the PSP_VPSS_QUEUE request is as follows.

- If the application issues PSP_VPSS_QUEUE request with the buffer size less than size of available statistics, then driver will return error.

The behavior of submit call for the PSP_VPSS_DEQUEUE request is as follows.

- If the application issues PSP_VPSS_DEQUEUE request without performing PSP_VPSS_QUEUE operation then driver will return error.
- If the application issues PSP_VPSS_DEQUEUE request after performing PSP_VPSS_QUEUE operation, the driver will block the PSP_VPSS_DEQUEUE request for 150ms. If statistics are available before time out occurs, then driver will return the statistics to the application otherwise error will be returned.
- When application puts a buffer in a queue for any of AF or AEW Channels, the driver will automatically enabled & start capturing frame. After completion of a frame the driver will copy the data into queued buffer and if another buffer is available in the in AF or AEW ACTIVE queue, interrupt will remain enabled otherwise it will be disabled. Enqueue/Dequeue will follow the FIFO mechanism

3.2 Functions for Auto Exposure/ Auto White Balance Driver

This section lists the functions available in the PSP module.

3.2.1 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 Auto Exposure/ Auto White Balance channel. GIO_create () populates static settings in driver object, formally creates/registers driver entry point with DSP/BIOS. Along with Auto Exposure/Auto White Balance channel, CCDC Channel must be opened and configured.

Note:

- It is assumed that the application has initialized the memory pool from which it desires the allocation to happen.
- The Auto Exposure/Auto White Balance should not be enabled for Foveon formatted input

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] Channel Type
attrs	[IN] pointer to GIO_attrs structure

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

Example:

```
void AEW_open()
{
      GIO_Handle   aewChan;
      PSP_H3AChannelType type;
      Int gioStatus;
      /* Configure CCDC Engine */
      /* open a logical channel */
      type= PSP_H3A_AEW;
      aewChan =GIO_create("/H3A",IOM_INOUT,&gioStatus,&type,&gioAttrs);
}
deviceName is passed as H3A. Instance is declared in the configuration
file. Type is type of the channel( AF or AEW).
```

3.2.2 GIO_delete

Int GIO_delete	**(GIO_Handle gioChan)**

This function deletes the driver channel.

Parameters:

gioChan	[IN] GIO Handle for the channel

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

Example:

```
/* close the AEW channel */
GIO_delete(afChan);
/* Close CCDC Channel */
GIO_delete(CcdcHandle);
```

Note:

- CCDC Handle must be deleted. If handle is not deleted, CCDC Engine will remain enabled. If the program is executed for the next time then Auto Exposure/ Auto White Balance may generate invalid statistics.

3.2.3 GIO_control

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

This function handles the IOCTLs for the Auto Exposure /Auto White Balance 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 FIFO is not enabled or if the mode is not supported

Example:

```
#define WIN_VT_COUNT        (3u)
#define WIN_HZ_COUNT        (3u)
#define H3A_AEW_BUFFER_SIZE
(WIN_VT_COUNT * WIN_HZ_COUNT * PSP_AEW_WIN_SIZE)
/* configure the logic channel */
```

```
static PSP_AEWParams params =
{
    PSP_AEW_ALAW_DISABLE,   /* A Law Status     */
    120u,                   /* Saturation limit */
    {
        10u,                /* width    */
        10u,                /* height   */
        4u,                 /* horizontal start */
        4u,                 /* vertical start   */
        WIN_HZ_COUNT,       /* horizontal count */
        WIN_VT_COUNT,       /* vertical count   */
        4u,                 /* horizontal line increament   */
        4u,                 /* vertical line increament     */
    },
    {
        10u,                /* black window height          */
        4u,                 /* black window vertical start  */
    }
};
GIO_control(aewChan, PSP_H3A_IOCTL_SET_PARAM, &params);
```

Note:
Some of the points to be taken into consideration while configuring CCDC are:

- Video Port should be enabled.
- AEW does not support for the Foveon sensor. The input must be in bayer format.

Some of the points to be taken into consideration while configuring Auto Exposure/ Auto White Balance are:

- The width and height of the windows must be an even number.
- Sub-sampling windows can only start on even numbers.
- The minimum width of the AE/AWB windows must be 6 pixels.

3.2.4 GIO_submit

PSP_Result GIO_submit	**(GIO_Handle**	**gioChan,**
	Uns	**cmd,**
	Ptr	**bufp,**
	Uns	***pSize,**
	GIO_AppCallback	***appCallback)**

This function handles the IOCTLs for the H3A driver.

Parameters:

gioChan	GIO Handle for the channel
cmd	Specified mini-driver command.
bufp	Pointer to data structure of buffer data
pSize	Size of data buffer structure.
appCallback	Pointer to call back function.

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

Example:

```
GIO_Handle afChan;
Int bufferSize;
PSP_H3ABufferbuffer;
PSP_H3ABuffer dequebuffer;

/* Size of the buffer Data Structure*/
bufferSize = sizeof(PSP_H3ABuffer);
buffer.size = H3A_AEW_BUFFER_SIZE;

/* For proper cache operation,
   Buffer size should be multiple of 128 bytes */
buffer.ramIpAddr  = (void *)MEM_alloc(0, buffer.size,(128u));

/* submit the empty buffer */
GIO_submit(aewChan, PSP_VPSS_QUEUE, &buffer, &bufferSize, NULL);

/AEW Engine is enabled */
/* Enable CCDC Engine */
GIO_submit(aewChan, PSP_VPSS_DEQUEUE, &dequebuffer, &bufferSize, NULL);
/* Invalidate the cache */
```

Note :

- The input to the Auto Exposure/ Auto White Balance module is the video port interface of the CCD Controller, In order to guarantee that data from the CCD Controller is not missed, the Auto Exposure/ Auto White Balance should be enabled prior to the CCD Controller.
- Cache must be invalidated to ensure that data is read from SDRAM/DDRAM.
- For proper cache operation buffer size should be multiple of 128 bytes (due to cache line size).

The behavior of submit call for the PSP_VPSS_QUEUE request is as follows.

- If the application issues PSP_VPSS_QUEUE request with the buffer size less than size of available statistics, then driver will return error.

The behavior of submit call for the PSP_VPSS_DEQUEUE request is as follows.

- If the application issues PSP_VPSS_DEQUEUE request without performing PSP_VPSS_QUEUE operation then driver will return error.
- If the application issues PSP_VPSS_DEQUEUE request after performing PSP_VPSS_QUEUE operation then driver will block the PSP_VPSS_DEQUEUE request for some predefined time. If statistics are available before time out occurs, then driver will return the statistics to the application otherwise driver will return error.

3.3 Data Structures Configuration defines for H3A Engine

The file **psp_h3a.h** has the **_PSP_H3ABuffer** data structure that is passed for configuration of paxels. Region of two dimensional blocks of data in an image is referred to as a “Paxel” for the case of AF. The parameters of the structure are explained below.

1) Buffer data structure

Parameter	Description

nodeEntry	This is node for the queue entry. This is of the variable of type struct _ PAL_OsListNodeHeader. This structure is used to keep track of next and previous elements in the queue.
ramIpAddr	Buffer address to store data. The statistics will be generated in this buffer. This address must be 64 bits aligned.
timeStamp	Time at which statistics are generated. This will expressed in terms of milliseconds.
Size	This is size of enqueued buffer. This size should be configured as Size for AF = Horizontal Count of Paxel * Vertical Count of Paxel * Size of Paxel (48 bytes). Size for AEW = Horizontal Count of window * Vertical Count of window * Size of window (18 bytes).
buffStatus	Status of statistics generated in the buffer. This field is of type enum _PSP_PSP_H3ABufferStatus.

3.4 Data Structures Configuration defines for Auto Focus Engine

The file **psp_af.h** has the **_PSP_AFPaxel** data structure that is passed for configuration of paxels. The parameters of the structure are explained below.

1) Paxel data structure

Parameter	Description
Width	Width of the paxel. The minimum width is expected to be 6.
Height	Height of the paxel. Range is 2 to 256
hzStart	Horizontal Start of the paxel. . Horizontal start must be equal to or greater than (IIR Filter Horizontal Start Position +2). This value must be even. Range is 2 to 4094.
vtStart	Vertical Start of the paxel. Range is 0 to 4095.
hzCnt	No of paxels in the horizontal direction. Range is 2 to 36.
vtCnt	No of paxels in the horizontal direction. Range is 1 to 128.
lineIncr	Number of lines to increment in a Paxel. If all the lines are to be processed, this field should be set to zero

The file **psp_af.h** has the **_PSP_AFHmf** data structure is used to configure the hmf filter. The parameters of the structure are explained below:

2) Horizontal Median Filter data Structure

Parameter	Description
enable	Status of Horizontal Median Filter. This field is of type enum

	_PSP_AF_HMF_law.
threshold	Horizontal Filter Threshold. Maximum value is 255.

The file **psp_af.h has** the **_PSP_AFIir** data structure that is passed for configuration of IIR Filters. The parameters of the structure are explained below:

3) IIR filter data structure

Parameter	Description
hzStarPos	IIR Filter Horizontal Start Position. Range is 0 to 4094. This value must be even.
coeffSet0[PSP_AF_NUMBER_OF_COEF]	Coefficients of the IIR Filter 0. Each coefficient is 12 bits wide with 6 bits of the decimal precision. Coefficients are signed.
coeffSet1[PSP_AF_NUMBER_OF_COEF]	Coefficients of the IIR Filter 1. Each coefficient is 12 bits wide with 6 bits of the decimal precision. Coefficients are signed.

The file **psp_af.h** has the **_PSP_AFParams** data structure that is passed to GIO_control to configure Auto Focus channel. The parameters of the structure are explained below:

4) Parameter data structure for Auto Focus engine

Parameter	Description
enable	Enumeration value that is used to enable or disable A law. This filed is of type enum _PSP_AF_Alaw. The A-law conversion routine compresses the 10-bit value to an 8-bit value.
rgbPos	RGB Position. This field is of type enum _PSP_AF_rgbpos. This specifies the colour pattern.
hmfConfig	Variable of the type structure _PSP_AFHmf. It store configuration for HMF Filter.
iirConfig	Variable of the type structure _PSP_AFIir. It store configuration for IIR Filters.
paxelConfig	Variable of the type structure _PSP_AFPaxel. It store configuration for paxels.
mode	Accumulator mode. This is of type enum PSP_AF_mode.

3.5 Data Structures Configuration defines for Auto Exposure / Auto White Balance

The file **psp_aew.h** has the **_PSP_AEWWindow** data structure that is passed for configuration of windows. Region of two dimensional blocks of data in an image is referred to as a “Windows” for the case of AEW. The parameters of the structure are explained below.

1) Window Data structure

Parameter	Description
width	Width of the window. The minimum width must be 6.This value must be even.
height	Height of the window. Range is 2 to 256. This value must be even.
hzStart	Sets the horizontal start for the window. Range is 0 to 4095.
vtStart	Sets vertical start of the window. Range is 0 to 4095.
hzCnt	Number of windows in horizontal direction. Range is 2 to 36.
vtCnt	Number of windows in vertical direction. Range is 1 to 128.
hzLineIncr	Sets horizontal distance between sub-samples within a window
vtLineIncr	Sets vertical distance between sub-samples within a window

The file **psp_aew.h** has the **_PSP_AEWBlkWindow** data structure is used to configure the HMF filter. The parameters of the structure are explained below:

2) Black window data structure

Parameter	Description
height	Height of the window.
vtStart	Sets the vertical start for the black window.

The file **psp_aew.h** has the **_PSP_AEWParams** data structure that is passed to GIO_control to configure Auto Exposure / Auto White Balance channel. The parameters of the structure are explained below:

3) Params structure for Auto Exposure/Auto White Balance engine

Parameter	Description
enable	Enumeration value that is used to enable or disable AEW Engine. This filed is of type enum _PSP_AEW_Alaw.
satLimit	Saturation limit. Maximum value is 1023.
winConfig	Variable of the type structure _PSP_AEWWindow .It stores configuration for window.
blkWinConfig	Variable of the type structure _PSP_AEWBlkWindow. It stores configuration for black window.

3.6 Symbolic Constants and Enumerated Data Types

This section summarizes all the symbolic constants specified as either #define macros and/or enumerated C data types. Described alongside the macro or enumeration is the semantics or interpretation of the same in terms of what value it stands for and what it means.

Group or Enumeration Class	Symbolic Constant Name	Description or Evaluation
Macro	PSP_AEW_WIN_SIZE	It describes the packet size for the window and should be 18.
Macro	PSP_AF_NUMBER_OF_COEF	The value should be 11.
Macro	PSP_AF_PAXEL_SIZE	It describes the packet size for the paxel and should be 48.
enum _PSP_H3AChannelType	PSP_H3A_AF	The application should pass this value to create Auto Focus channel.
enum _PSP_H3AChannelType	PSP_H3A_AEW	The application should pass this value to create Auto exposure / white balance channel.
enum _PSP_H3ABufferStatus	PSP_H3A_BUFFER_DATA_VALID	This status value indicates that the statistics contained in buffer are valid.
enum _PSP_H3ABufferStatus	PSP_H3A_BUFFER_DATA_CORRUPTED	This status value indicates that the statistics contained in buffer are corrupted.
enum _PSP_H3AIoctlCmd	PSP_H3A_IOCTL_SET_PARAM=128	This control command is used to set the parameters. This ioctl will configure hardware as per

		parameters passed by the user.
enum _PSP_H3AIoctlCmd	PSP_H3A_IOCTL_GET_PARAM	This control command is used to get the parameters. This ioctl will return the hardware configuration.
enum _PSP_H3AIoctlCmd	PSP_H3A_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.
enum _PSP_H3A_DEVICE_STATE	PSP_H3A_DELETED	This state indicates that H3A device is deleted.
enum _PSP_H3A_DEVICE_STATE	PSP_H3A_CREATED	This state indicates that H3A device is created.
enum _PSP_H3A_DEVICE_STATE	PSP_H3A_OPENED	This state indicates that H3A device is in use by the channel.
enum _PSP_H3A_DEVICE_STATE	PSP_H3A_CLOSED	This state indicates that H3A device is closed.
enum _PSP_AF_Alaw	PSP_AF_ALAW_DISABLE=0	This value will enable ALAW for AF Engine.
enum _PSP_AF_Alaw	PSP_AF_ALAW_ENABLE=1	This value will disable ALAW for AF Engine.
enum _PSP_AF_HMF_law	PSP_AF_HMF_DISABLE=0	Enables Horizontal Median Filter.
enum _PSP_AF_HMF_law	PSP_AF_HMF_ENABLE=1	Disables Horizontal Median Filter.
enum _PSP_AF_mode	PSP_AF_ACCUMULATOR_SUMMED=0	Summed mode of the AF

		Accumulator.
enum _PSP_AF_mode	PSP_AF_ACCUMULATOR_PEAK=1	Peak mode of the AF Accumulator.
enum _PSP_AF_rgbpos	PSP_AF_GR_GB_BAYER=0	This value indicates GR and GB as Bayer pattern.
enum _PSP_AF_rgbpos	PSP_AF_RG_GB_BAYER=1	This value indicates RG and GB as Bayer pattern.
enum _PSP_AF_rgbpos	PSP_AF_GR_BG_BAYER=2	This value indicates GR and BG as Bayer pattern.
enum _PSP_AF_rgbpos	PSP_AF_RG_BG_BAYER=3	This value indicates RG and BG as Bayer pattern.
enum _PSP_AF_rgbpos	PSP_AF_GG_RB_CUSTOM=4	This value indicates GR and RB as custom pattern.
enum _PSP_AF_rgbpos	PSP_AF_RB_GG_CUSTOM=5	This value indicates RB and GG as custom pattern.
enum _PSP_AEW_Alaw	PSP_AEW_ALAW_DISABLE=0	This value will enable ALAW for AEW Engine.
enum _PSP_AEW_Alaw	PSP_AEW_ALAW_ENABLE=1	This value will disable ALAW for AEW Engine.

Chapter 4

EXAMPLE APPLICATIONS

This section describes the example applications that are included in the package. These sample applications can be used for quick demonstration. User will get benefit by using these samples as reference source code in developing new applications.

4.1 Writing Applications for H3A

This section provides guidance to user for writing their own application for H3A driver

4.1.1 File Inclusion

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

1. **psp_h3a.h**

   This file contains the interfaces, data types and symbolic definitions that are needed by the application to utilizes the services of H3A device driver.

2. **psp_af.h**

   This file contains the interfaces, data types and symbolic definitions that are needed by the application to utilizes the services of AF module of H3A device driver.

3. **psp_aew.h**

   This file contains the interfaces, data types and symbolic definitions that are needed by the application to utilizes the services of AEW module of H3A device driver.

4.1.2 Driver Initialization

To use the H3A device driver, a device entry must be added and configured in the DSP/BIOS configuration tool.

To have H3A device driver included in the application, corresponding TCI file have to be included in BIOS TCF i.e. "*dm6437_h3a0.tci*" must be included in BIOS TCF file of the application. This file can be found in video sample directory.

The H3A driver initialization in BIOS TCF looks like the following:

```
bios.UDEV.create("H3A0");
bios.UDEV.instance("H3A0").fxnTable = prog.extern("H3AMD_FXNS");
bios.UDEV.instance("H3A0").fxnTableType = "IOM_Fxns";
```

4.1.3 Dependent Projects/Libraries

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

- H3A
- VPFE
- I2C
- Previwer
- 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 AF Sample Application

4.2.1 Introduction

This “psp_bios_af_st_example.pjt” application demonstrates Auto Focus functionality of H3A. This application collects statistics of a frame generated by Auto Focus Engine . Before running the sample application of Auto Focus 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

Build the VPFE library with PSP_VIDEO_PATH_ENABLE Macro enabled in psp_vpfe.h.
The sample application project file is located in the
<root>packages\ti\sdo\pspdrivers\system\dm6437\bios\evmDM6437\video\sample\build\h3a
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 studio.
After running the sample application of Auto Focus, execute the previewer on the fly application to ensure output image is still stable.

4.3 AEW Sample Application

4.3.1 Introduction

This “psp_bios_aew_st_example.pjt” application demonstrates Auto Exposure / Auto White Balance functionality of H3A. This application collects statistics of a frame generated by Auto Exposure / Auto White Balance Engine.

Before running the sample application of Auto Exposure/Auto White Balance run the previewer on the fly application to ensure that sensor is configured correctly and input image is stable.

4.3.2 Building the Application

Build the VPFE library with PSP_VIDEO_PATH_ENABLE Macro enabled in psp_vpfe.h.
The sample application project file is located in the *<root>packages\ti\sdo\pspdrivers\system\dm6437\bios\evmDM6437\video\sample\build\h3a* 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 studio.
After running the sample application of Auto White Exposure/Auto White Balance, execute the previewer on the fly application to ensure output image is still stable.

Chapter 5

H3A Performance Results

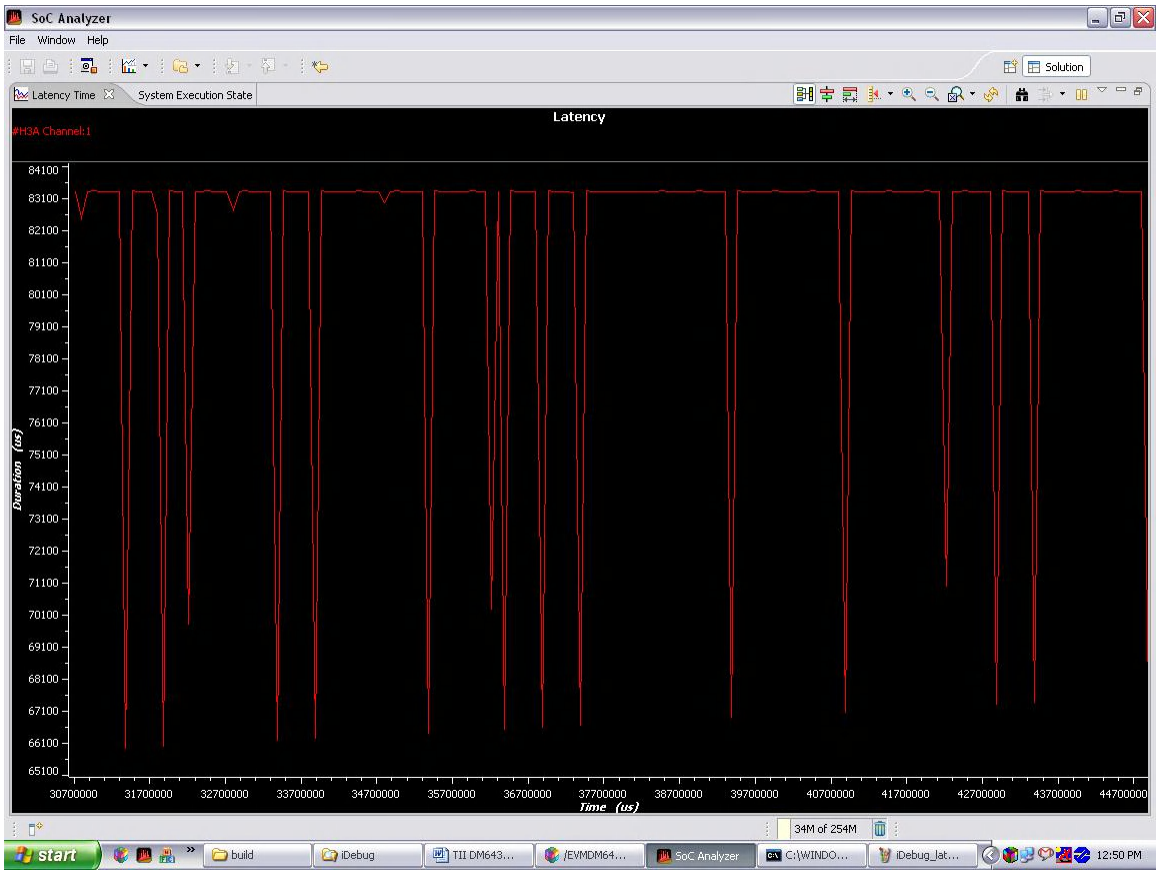

Figure 3 Latency graph of H3A – AEW

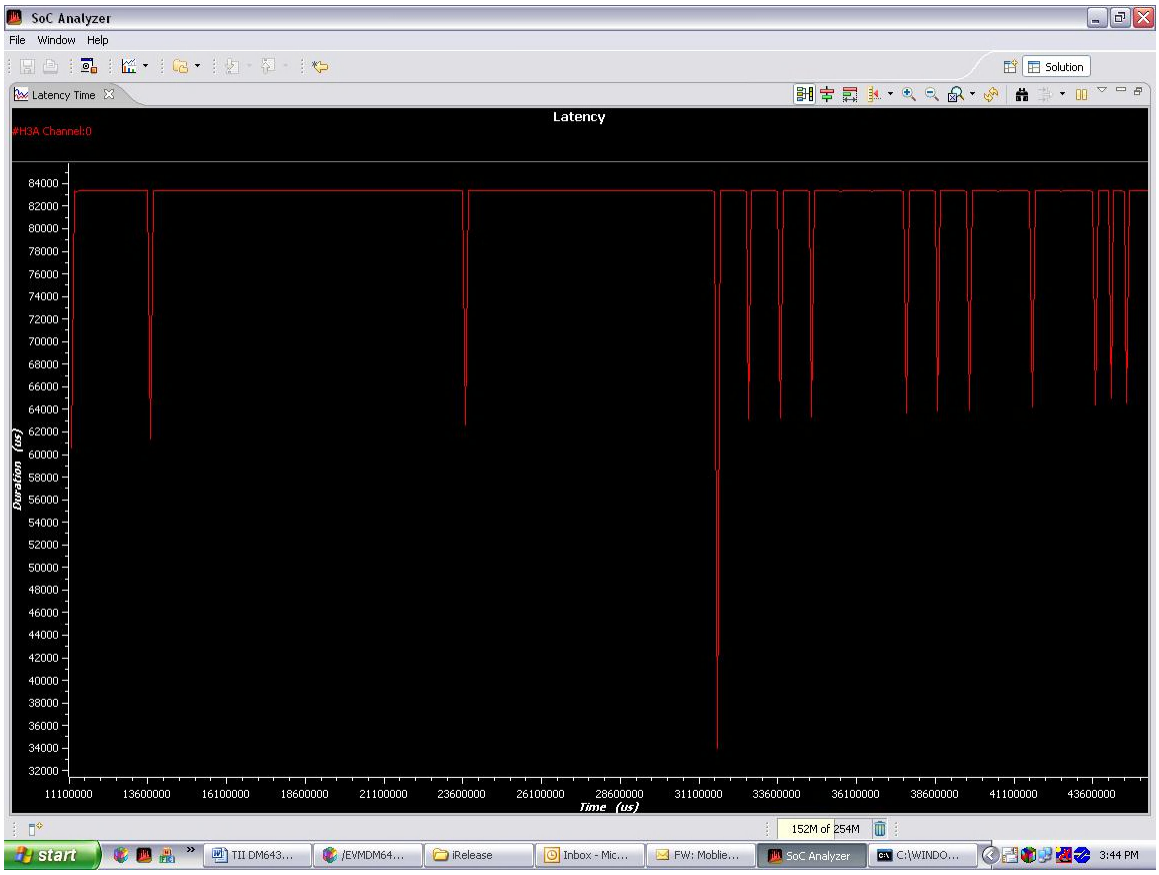

Figure 4 Latency graph of H3A – AF

Note: The Driver Performance Characteristics can be included once testing is done on DM6437 SOC. The graphs are taken for H3A for both AEW and AF in interrupt mode.