TII DM6437 VPSS Drivers
Previewer API Specifications

Release 1.10.00

January 14, 2008

Table of Contents

TABLE OF FIGURES

Revision History

Date	Version	Changes	Author
Oct 10, 2006	Draft 0.01	Created	EI2
Oct 11, 2006	Issue 1.00	Updated for technical review comments	EI2
Nov 19.2006	Issue 1.01	Updated after incorporating Changes found during coding	EI2
Nov 20.2006	Pre-silicon Release 0.3.0	Release to TI	EI2
Nov 30,2006	Post-silicon Release 0.3.0	Release to TI	EI2
Dec 4,2006	Post-silicon Release 0.3.0	Unused IOCTLs are removed, as CCDC is using LLC functionality directly.	EI2
Dec 21,2006	Post-silicon Release 0.3.0	Updated for dark frame capture feature	EI2
June 22, 2007	1.00.01	Updated for 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	EI2
November 29, 2007	1.00.04	PSP merge package changes - directory structure changes	Sivaraj R
January 14, 2008	1.00.05	PSP_PREVIEWER_IOCTL_SET_SEM_TIMEOUT IOCTL added	Sivaraj R

Overview

Purpose and Scope

This document provides APIs for the proposed driver for the Previewer 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 Previewer driver shall be DM6437_previewer. Hence the name of the top level files which will directly interact with application shall be “dda_previewerIOM.h”. These above files will interact with the “dda_previewer.c”. Thereafter the “dda_previewer.c” will interact with the “ddc_previewer.c”. Finally, the files related to hardware block will be referred as the “llc_previewer.c”.

Architecture

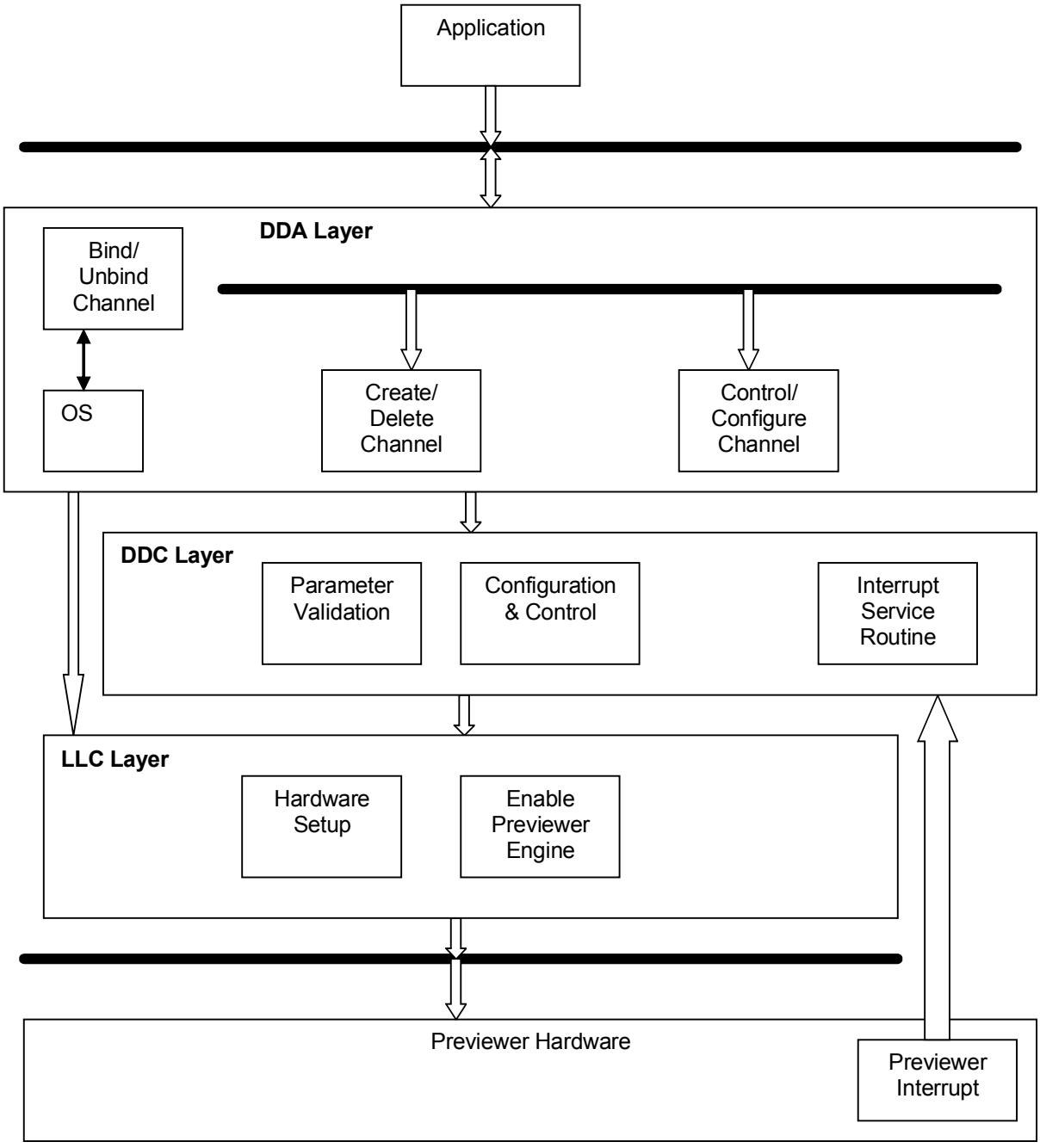

Figure 1. Top-Level Block Diagram of Previewer

The previewer 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

It validates the parameters send by the application. It also handles the ISR. The previewer interrupt is generated after the completion of previewer data transfer.

LLC Layer

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

Critical Features and Implementation

None.

2. Application Level APIs

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

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

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 "/DM6437_previewer.
- **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. In the case of previewer, it will be IOM_INOUT.
- **status**
  If the status parameter is non-NULL, a status value is placed at the address specified by the status parameter.
- **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.
  In case of previewer, pointer of PSP_previewerChannelCreateMode type will be passed. This gives information to driver that source for previewer is SDRAM or CCDC.
- **attrs**
  The attrs parameter is a pointer to a structure of type GIO_Attrs.

Description

Open a logical channel. Multiple open will not be supported by the Previewer driver.
If source is CCDC, at that time, configuration for this channel will be done from CCDC, and application will be able to call only GIO_delete. (GIO_control will not be called from application in this case.)

```
/**
 * \brief PSP_previewerChannelSource
 *
 * This structure is used as a member element of _PSP_previewerChannelCreateMode
 * strcture.
 */
typedef struct _PSP_previewerChannelSource
{
   Uint8 source;
   /**< Value: PSP_PREVIEWER_CHANNEL_CCDC or PSP_PREVIEWER_CHANNEL_SDRAM
*/
}PSP_previewerChannelSource;

/**
 * \brief PSP_previewerChannelCreateMode
 *
 * This strcture is filled at the time of channel creation.
 */
```

```
typedef struct _PSP_previewerChannelCreateMode
{
   PSP_previewerChannelSource chanSource;
   /**< Channel Source */
   Int32 segId;
   /**< Segment Id passed by applicaion, will be used to allocate memory */

}PSP_previewerChannelCreateMode;

/**< previewer source      */
#define PSP_PREVIEWER_CHANNEL_CCDC
#define PSP_PREVIEWER_CHANNEL_SDRAM
```

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, 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 Previewer control requests are located in section 3 of this document.
Return Value
IOM_COMPLETED on success and negative value if error.

3. Previewer module Controls

3.1 PSP_PREVIEWER_IOCTL_SET_PARAMS

Name

PSP_PREVIEWER_IOCTL_SET_PARAMS – sets the Previewer hardware parameters associated with this channel

Synopsis

int GIO_control(GIO_Handle gioChan, int cmd, PSP_previewerParams *previewerParam);

Arguments

- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_IOCTL_SET_PARAMS
- **previewerParam**
  Pointer to the PSP_previewerParams [IN]

Description

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

```
typedef struct _PSP_previewerParams
{
   Uint32                  features;
   /**< Set of features enabled */
   PSP_previewerSize               sizeParam;
   /**< size parameters */
   PSP_previewerWhiteBalance       whiteBalanceParam;
   /**< white balancing parameters */
   PSP_previewerBlackAdj           blackAdjParam;
   /**< black adjustment parameters */
   PSP_previewerRgbBlending        rgbBlendingParam;
   /**< rgb blending parameters */
   PSP_previewerRgb2ycbcrCoeffs    rgb2ycbcrParam;
   /**< rgb to ycbcr parameters */
   PSP_previewerCfaCoeffs          cfaCoeffsParam;
   /**< CFA coefficients   */
   PSP_previewerGammaCoeffs        gammaCoeffsParam;
   /**< gamma coefficients */
   PSP_previewerNoiseFilterCoeffs  noiseFilterCoeffsParam;
   /**< noise filter coefficients  */
   PSP_previewerChromaSuppression  chromaSuppressionParam;
   /**< chroma suppression coefficients */
   PSP_previewerOutPixelOrder      outPixelOrderParam;
   /**< output pixel format */
   Uint32                  lumaEnhance[PSP_PREVIEWER_LUMA_TABLE_SIZE];
   /**< luma enhancement coeffs */
   Ptr                     darkFrameAddr;
   /**< dark frame address */
   Uint16                  darkFrameOffset;
   /**< dark frame offset */
   /**< dark frame address */
   Int16                   hmfThreshold;
   /**< horizontal median filter threshold */
```

```
    Uint8                   contrast;
    /**< contrast */
    Uint8                   brightness;
    /**< brightness */
    Uint8                   downSampleRate;
    /**< down sampling rate for averager */
    Uint8                   lensShadingShift;
    /**< number of bits to be shifted for lens shading  */
} PSP_previewerParams

/* features field can be OR'ed with the following flags */
#define PSP_PREVIEWER_INVERSE_ALAW
#define PSP_PREVIEWER_HMF
#define PSP_PREVIEWER_NOISE_FILTER
#define PSP_PREVIEWER_CFA
#define PSP_PREVIEWER_GAMMA
#define PSP_PREVIEWER_LUMA_ENHANCE
#define PSP_PREVIEWER_CHROMA_SUPPRESS
#define PSP_PREVIEWER_DARK_FRAME_SUBTRACT
#define PSP_PREVIEWER_LENS_SHADING

/* PSP_previewerSize has following possible values */
typedef struct _PSP_previewerSize
{
    Uint16  inPitch;
    /**< line offset of input image - used when source is SDRAM */
    Uint16  outPitch;
    /**< line offset of output image */
    Uint8 sph;
    /**< start pixel horizontal */
    Uint8 eph;
    /**< end pixel horizontal */
    Uint8 slv;
    /**< start line vertical */
    Uint8 elv;
    /**< end line vertical */
    Uint8   pixelSize;
    /**< pixel size of the image in terms of bits - used when source is SDRAM */

}PSP_previewerSize;

/* The pixsize field in prev_size_params has 2 possible values */
/**< pixel width of 8 bits   */
#define PSP_PREVIEWER_INWIDTH_8BIT
/**< pixel width of 10 bits   */
#define PSP_PREVIEWER_INWIDTH_10BIT


/*structure for white balancing parameters*/
typedef struct _PSP_previewerWhiteBalance
{
    Uint8  position[PSP_PREVIEWER_WBA_POS_MAX][ PSP_PREVIEWER_WBA_POS_MAX];
    /**< 16 position, out of 4 values */
    Uint16  wbDgain;
   /**< white balance common(digital) gain */
    Uint8  wbGain[PSP_PREVIEWER_WB_GAIN_MAX];
```

```
    /**< individual color gains */
}PSP_previewerWhiteBalance;

/*number of coefficients for while balanced gain    */
#define PSP_PREVIEWER_WB_GAIN_MAX     4

/*structure for black adjustment for colors*/
typedef struct _PSP_previewerBlackAdj
{
    Int8 redAdj;
    /**< black adjustment offset for red color */
    Int8 greenAdj;
    /**< black adjustment offset for green color */
    Int8 blueAdj;
    /**< black adjustment offset for blue color */
}PSP_previewerBlackAdj;

/*structure for RGB2RGB blending parameters*/
typedef struct _PSP_previewerRgbBlending
{
    Int16 blending[PSP_PREVIEWER_RGB_MAX][PSP_PREVIEWER_RGB_MAX];
    /**< color correlation 3x3 matrix */
    Int16 offset[PSP_PREVIEWER_RGB_MAX];
    /**< color correlation offsets */
}PSP_previewerRgbBlending;

/**< matrix size for RGB2RGB blending        */
#define PSP_PREVIEWER_RGB_MAX         3

/*structure RGB2YCbCr parameters*/
typedef struct _PSP_previewerRgb2ycbcrCoeffs
{
    Int16 coeff[PSP_PREVIEWER_RGB_MAX][PSP_PREVIEWER_RGB_MAX];
   /**< color conversion gains in 3x3 matrix */
    Uint16 yOffset;
    /**< y color conversion offsets */
    Int8 cbOffset;
    /**< Cb color conversion offsets */
    Int8 crOffset;
    /**< Cr color conversion offsets */
}PSP_previewerRgb2ycbcrCoeffs;

/*structure for CFA coefficients*/
typedef struct _PSP_previewerCfaCoeffs
{
    Uint8    hThreshold;
   /**< horizontal threshold */
    Uint8    vThreshold;
    /**< vertical threshold */
   Int32 coeffs[PSP_PREVIEWER_CFA_COEFF_TABLE_SIZE];
   /**< cfa coefficients */
}PSP_previewerCfaCoeffs;

/*structure for Gamma Coefficients*/
typedef struct _PSP_previewerGammaCoeffs
{
```

```
    Uint8 red[PSP_PREVIEWER_GAMMA_TABLE_SIZE];
    /**< table of gamma correction  values for red */
    Uint8 green[PSP_PREVIEWER_GAMMA_TABLE_SIZE];
    /**< table of gamma correction  values for green */
    Uint8 blue[PSP_PREVIEWER_GAMMA_TABLE_SIZE];
    /**< table of gamma correction  values for blue */
}PSP_previewerGammaCoeffs;

/*structure for Noise Filter Coefficients*/
typedef struct _PSP_previewerNoiseFilterCoeffs
{
    Uint8 noise[PSP_PREVIEWER_NOISE_FILTER_TABLE_SIZE];
    /**< noise filter table */
    Uint8 strength;
    /**< to find out weighted average */
}PSP_previewerNoiseFilterCoeffs;

/*structure for Chroma Suppression*/
typedef struct _PSP_previewerChromaSuppression
{
    Uint8 hpfy;
    /**< whether to use high passed version of Y or normal Y */
    Int8 threshold;
    /**< threshold for chroma suppress */
    Uint8 gain;
    /**< chroma suppression gain */

}PSP_previewerChromaSuppression;

/* table size for different modules  */
#define PSP_PREVIEWER_LUMA_TABLE_SIZE           128
#define PSP_PREVIEWER_GAMMA_TABLE_SIZE          1024
#define PSP_PREVIEWER_CFA_COEFF_TABLE_SIZE      576
#define PSP_PREVIEWER_NOISE_FILTER_TABLE_SIZE   256

/* Output pixel format can be one of these values */
typedef enum _PSP_previewerOutPixelOrder
{
    PSP_PREVIEWER_PIXELORDER_YCBYCR,
    /**< LSB Y0 Cb0 Y1 Cr0 MSB */
    PSP_PREVIEWER_PIXELORDER_YCRYCB,
    /**< LSB Y0 Cr0 Y1 Cb0 MSB */
    PSP_PREVIEWER_PIXELORDER_CBYCRY,
    /**< LSB Cb0 Y0 Cr0 Y1 MSB */
    PSP_PREVIEWER_PIXELORDER_CRYCBY
    /**< LSB Cr0 Y0 Cb0 Y1 MSB */
}PSP_previewerOutPixelOrder;
```

Return Value

IOM_COMPLETED on success and negative value if an error occurred.

3.2 PSP_PREVIEWER_IOCTL_GET_PARAMS

Name

PSP_PREVIEWER_IOCTL_GET_PARAMS – get the Previewer hardware parameters associated with this logic channel.

Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_previewerParam * previewerParam);

Arguments
- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_IOCTL_GET_PARAMS
- **previewerParam**
  Pointer to the PSP_previewerParams structure [OUT]

Description
This ioctl is used to get the Previewer hardware settings.
Return Value
IOM_COMPLETED on success and negative value if error.

PSP_PREVIEWER_IOCTL_GET_STATUS

Name
PSP_PREVIEWER_IOCTL_GET_STATUS – get the current status of the hardware

Synopsis
int GIO_control (GIO_Handle gioChan, int cmd, PSP_previewerStatus * previewerStatus);
Arguments
- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_IOCTL_GET_STATUS
- **previewerStatus**
  Pointer to the previewerStatus structure [OUT]

Description
This ioctl is used to check the current status of the Previewer hardware.
The PSP_previewerStatus structure is defined as shown below:

```
typedef struct _PSP_previewerStatus
{
   Uint8 channel_status;
   /**< value can be PREVIEWER_CHANNEL_BUSY or PREVIEWER_CHANNEL_FREE */
}PSP_previewerStatus;

/**< previewer channel busy status   */
#define PSP_PREVIEWER_CHANNEL_BUSY
#define PSP_PREVIEWER_CHANNEL_FREE
```

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

PSP_PREVIEWER_IOCTL_GET_DARK_FRAME_STATUS

Name
PSP_PREVIEWER_IOCTL_GET_DARK_FRAME_STATUS – get the Dark Frame Subtract Fail Status and also clear it.
Synopsis
int GIO_control(GIO_Handle gioChan, int cmd,
PSP_previewerDarkFrameStatus * previewerDarkFrameStatus);

Arguments

- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_IOCTL_GET_DARK_FRAME_STATUS
- **previewerDarkFrameStatus**
  Pointer PSP_previewerDarkFrameStatus structure[OUT]

Description

This ioctl is used to check the Dark Frame Fail Status.
The PSP_previewerDarkFrameStatus structure is defined as shown below:

```
typedef struct _PSP_previewerDarkFrameStatus
{
   Uint8 darkFrameFrameStatus;
   /**< value can be PSP_PREVIEWER_DARK_FRAME_FAILED
       or PSP_PREVIEWER_DARK_FRAME_WORKING */
}PSP_previewerDarkFrameStatus;

/**< previewer dark frame subtract failure status */
#define PSP_PREVIEWER_DARK_FRAME_FAILED
#define PSP_PREVIEWER_DARK_FRAME_WORKING
```

Return Value

IOM_COMPLETED on success and negative value if an error occurred.

PSP_PREVIEWER_IOCTL_PREVIEW

Name

PSP_PREVIEWER_IOCTL_PREVIEW– to trigger previewer when source is SDRAM

Synopsis

int GIO_control(GIO_Handle gioChan, int cmd, PSP_preview *previewSize);

Arguments

- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_IOCTL_PREVIEW
- **previewSize**
  Pointer to the PSP_preview structure [IN]

Description

This ioctl is used to trigger previewer when source is SDRAM
The PSP_preview structure is defined as shown below:

```
typedef struct _PSP_preview
{
   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_preview;
```

Return Value

IOM_COMPLETED on success and negative value if an error occurred

PSP_PREVIEWER_IOCTL_GET_CROPSIZE

Name
PSP_PREVIEWER_IOCTL_GET_CROPSIZE- to get crop size
Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_previewerCropSize *previewerCropSize);
Arguments
- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_IOCTL_GET_CROPSIZE
- **previewerCropSize**
  Pointer to PSP_previewerCropSize structure [OUT]

Description
This ioctl is used to get crop size
The PSP_previewerCropSize structure is defined as shown below:

```
typedef struct _PSP_previewerCropSize
{
   Int32 hCrop;
   Int32 vCrop;
}PSP_previewerCropSize;
```

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

PSP_PREVIEWER_GET_INFO_FOR_CCDC

Name
PSP_PREVIEWER_GET_INFO_FOR_CCDC– to get the information required for CCDC continous mode
Synopsis
int GIO_control(GIO_Handle gioChan, int cmd, PSP_previewerGetInfoForCCDC *previewInfoForCCDC;
Arguments
- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_GET_INFO_FOR_CCDC
- **previewInfoForCCDC**
  Pointer to the PSP_previewerGetInfoForCCDC structure [IN]

Description
This structure is used to get info of previwer required by CCDC
The PSP_previewerGetInfoForCCDC structure is defined as shown below:

```
typedef struct _PSP_previewerGetInfoForCCDC
{
   Uint8 source;
   /**< interrupt source */
   Uint8 intNo;
   /**< interrupt number */
}PSP_previewerGetInfoForCCDC
```

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

PSP_PREVIEWER_IOCTL_SET_EXP

Name

PSP_PREVIEWER_IOCTL_SET_EXP - to set read request expand field

Synopsis

int GIO_control(GIO_Handle gioChan, int cmd, PSP_previewerReadReqExp *prevReadReqExpParam);

Arguments

- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_IOCTL_GET_CROPSIZE
- **prevReadReqExpParam**
  Pointer to PSP_previewerReadReqExp structure [IN]

Description

This ioctl is used to set read request expand field.
The PSP_previewerReadReqExp structure is defined as shown below:

```
typedef struct _PSP_previewerReadReqExp
{
   Int16 prevExp;
}PSP_previewerCropSize;
```

Return Value

IOM_COMPLETED on success and negative value if an error occurred

PSP_PREVIEWER_ SET_DARK_FRAME_CAPTURE

- **Name**

PSP_PREVIEWER_ SET_DARK_FRAME_CAPTURE - to set Dark Frame Capture field

- **Synopsis**

int GIO_control(GIO_Handle gioChan, int cmd, PSP_previewerDarkFrameCapture *prevDarkFrameCapture);

Arguments

- **gioChan**
  Handle to an instance of the device
- **request**
  PSP_PREVIEWER_ SET_DARK_FRAME_CAPTURE
- **prevDarkFrameCapture**
  Pointer to PSP_previewerDarkFrameCapture structure [IN]

Description

This ioctl is used to set Dark Frame Capture field.
The PSP_previewerDarkFrameCapture structure is defined as shown below:

```
typedef struct _PSP_previewerDarkFrameCapture
{
 Uint8 darkFrameState;
   /**< Value can be PSP_PREVIEWER_DARK_FRAME_CAPTURE_DISABLE or
    PSP_PREVIEWER_DARK_FRAME_CAPTURE_ENABLE  */
   Uint16 outPitch;
   /**< offset in dark image for each row. Value is only relevant when
    flag is PSP_PREVIEWER_DARK_FRAME_CAPTURE_ENABLE */
}PSP_previewerDarkFrameCapture;
```

Return Value

IOM_COMPLETED on success and negative value if an error occurred

PSP_PREVIEWER_ GET_DARK_FRAME_CAPTURE

PSP_PREVIEWER_ GET_DARK_FRAME_CAPTURE - to get Dark Frame Capture field

Synopsis

int GIO_control(GIO_Handle gioChan, int cmd, PSP_previewerDarkFrameCapture *prevDarkFrameCapture);

Arguments

gioChan

Handle to an instance of the device

request

PSP_PREVIEWER_ GET_DARK_FRAME_CAPTURE

prevDarkFrameCapture

Pointer to PSP_previewerDarkFrameCapture structure [IN]

Description

This ioctl is used to get Dark Frame Capture field.

The PSP_previewerDarkFrameCapture structure is defined as shown below:

```
typedef struct _PSP_previewerDarkFrameCapture
{
 Uint8 darkFrameState;
   /**< Value can be PSP_PREVIEWER_DARK_FRAME_CAPTURE_DISABLE or
    PSP_PREVIEWER_DARK_FRAME_CAPTURE_ENABLE  */
   Uint16 outPitch;
   /**< offset in dark image for each row. Value is only relevant when
    flag is PSP_PREVIEWER_DARK_FRAME_CAPTURE_ENABLE */
}PSP_previewerDarkFrameCapture;
```

Return Value

IOM_COMPLETED on success and negative value if an error occurred

PSP_PREVIEWER_IOCTL_SET_SEM_TIMEOUT

Name

PSP_PREVIEWER_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_ PREVIEWER_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.

PSP_prevCreate

Name
PSP_prevCreate - It creates a given previewer device at DDA layer.
Synopsis
PSP_Result PSP_prevCreate(,PSP_Handle* devHandle);
Arguments
- **devHandle**
  Void pointer passed by the GIO Layer to store the address of the port structure

Description
It creates a given previewer device at DDA layer.
It performs parameter validation and initializes DDA port structure.
Assign DDA port structure instance to devp.

Return Value
PSP_SOK on success and PSP error if an error occurred

PSP_prevOpen

Name
PSP_prevOpen - It opens Channel of the previewer device.
Synopsis
PSP_Result PSP_prevOpen(PSP_Handle *chanHandle,
  PSP_Handle devHandle,
  PSP_previewerChannelCreateMode *chanParams);
Arguments
- **chanHandle**
  previewer Channel Handle which will be used in subsequent ioctl calls
- **devHandle**
  pointer to the port structure
- **chanParams**
  Extra parameter, will be passed at DDC layer

Description
It opens Channel of the previewer device.
The function prepares channel of the driver instance for data transfers and returns an handle to the channel instance. It performs parameter validation and initialize DDA channel structure. It will update port structure. it will call function for interrupt mapping.
Return Value
PSP_SOK on success and PSP error if an error occurred

PSP_prevIoctl

Name
PSP_prevIoctl - It controls Channel of the previewer device.
Synopsis
PSP_Result PSP_prevIoctl(PSP_Handle chanHandle,
  PSP_previewerControlCmd cmd,
  Ptr cmdArg);
Arguments
- **chanHandle**
  previewer Channel Handle
- **cmd**

Operation to be performed

- **cmdArg**
  Provides additonal information related to the operation

Description
It controls Channel of the previewer device.
This function provides ioctl functionality for previewer. It will call IOCTL of DDC layer.
Return Value
PSP_SOK on success and PSP error if an error occurred

PSP_prevClose

Name
PSP_prevClose - It closes Channel of the previewer device.
Synopsis
PSP_Result PSP_prevClose(PSP_Handle chanHandle);
Arguments

- **chanHandle**
  previewer Channel Handle

Description
It closes Channel of the previewer device.
This function is used to close Channel handle for previewer.
Return Value
PSP_SOK on success and PSP error if an error occurred

PSP_prevDelete

Name
PSP_prevDelete - It deletes a given previewer device at DDA layer.
Synopsis
PSP_Result PSP_prevDelete(PSP_Handle devHandle);
Arguments

- **devHandle**
  pointer to DDA port structure instance

Description
It deletes a given previewer device at DDA layer.
This function is called last and is used for driver deletion. This function does the deletion of the previewer device instance.
Return Value
PSP_SOK on success and PSP error if an error occurred

PSP_prevGetPSPHandle

Name
PSP_prevGetPSPHandle - It returns PSP handle of channel object.
Synopsis
PSP_Result PSP_prevGetPSPHandle (PSP_Handle *chanHandle);
Arguments

- **chanHandle**
  handle to channel

Description
It returns PSP handle of channel object.

Return Value
PSP_SOK on success and PSP error if an error occurred

4. Usage Examples

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

Driver open and close

```
/* open a logical channel */
GIO_Handle      prevHandle;
PSP_previewerChannelCreateMode createMode;
createMode.Source = PSP_PREVIEWER_CHANNEL_SDRAM;
prevHandle = GIO_create("/DM6437_previewer ",IOM_INOUT,NULL,& createMode,&gioAttrs);
if(prevHandle == NULL)
{
        printf("open to Previewer channel failed.\n")
        exit(-1);
}
/* close logical channel */
GIO_delete(prevHandle);
```

Fetch previewing parameters

```
PSP_previewerParams params;
/* Fetch parameters from the previewer */
GIO_control(prevHandle, PSP_PREVIEWER_IOCTL_GET_PARAMS, &params);
```

Setup previewing parameters

```
PSP_previewerParams params;
/* setup the parameter here */
params.sizeParam.hSize = 720;
params sizeParam.vSize = 240;
        -
        -
        -
/* configure the previewer */
GIO_control(prevHandle, PSP_PREVIEWER_IOCTL_SET_PARAMS, &params);
```

Initiate previewing operation

```
PSP_preview  preview;
/**< address of the input buffer */
preview .inBuf = inBuffer;
/**< address of the output buffer */
preview .outBuf =outBuffer;
/**< input buffer size */
preview .inBufSize = inBufferSize;
/**< output buffer size */
preview .outBufSize = outBufferSize;
GIO_control(prevHandle, PSP_PREVIEWER_IOCTL_PREVIEW,&preview);
```

Get the status of channel

```
PSP_previewerStatus previewerStatus;
GIO_control(prevHandle, PSP_PREVIEWER_IOCTL_GET_STATUS,&previewerStatus);
if (previewerStatus.channelStatus == PSP_PREVIEWER_CHANNEL_BUSY)
{
}
else
{
}
```

Get the dark frame failure status

```
PSP_previewerDarkFrameStatus previewerDarkStatus;
GIO_control(prevHandle, PSP_PREVIEWER_IOCTL_GET_DARK_FRAME_STATUS , ,
&previewerDarkStatus);
if(previewerDarkStatus. darkFrameFailStatus == PSP_PREVIEWER_DARK_FRAME_FAILED)
{
}
else
{
}
```

Get the crop size

```
PSP_previewerCropSize cropSize;
GIO_control(prevHandle, PSP_PREVIEWER_GET_CROPSIZE, &cropSize);
```

Set the preview read request expand field.

```
PSP_previewerReadReqExp prevReadReqExpParam;
GIO_control(prevHandle, PSP_PREVIEWER_IOCTL_SET_EXP, &prevReadReqExpParam);
```