W99702 Sensor DSP Engine
Library API
Reference Guide

Contents

Introduction

In this document, it will describe two W99702 engines – Sensor DSP engine, and parts of video capture engine. And it will show you how to access these 2 engines through the following C library functions.

The library for these two engines :

- The library name is called “**libdsp.a**” and its header file is “**libdsp.h**”.

Supported C Library Functions Listing

- Global Controls
    - dspEnableDSPinterrupt
    - dspSetIRQHandler
    - dspGetInterruptStatus
    - dspResetDSPengine
    - dspInitialization
    - dspEnableDSPfunc
    - dspDisableAllDSPfunc
    - dspGetDSPfuncStatus
    - dspSetSensorInterface
    - dspGetBayerRawData
    - dspSetSubsampling
    - dspSetCroppingWnd
    - dspSetTimingControl
    - dspSetSubWindow
- Black Level Compensation
    - dspSetBLCwnd
    - dspSetBLCMode
    - dspUpdateUserBlackLevels
    - dspGetDetectedBlackLevel
- False color suppression
    - dspFalseColorSupp
- High color suppression
    - dspHSS
- Auto white balance and Auto exposure
    - dspSetSceneMode
- White Balance Control
    - dspSetAWBcontrol
    - dspUpdateAutoWhiteBalance
    - dspUpdateWhiteBalance
    - dspGetColorGains
    - dspGetAWBstats
    - dspGetAWBstats_sw
    - dspGetWOcounts
- Exposure Control
    - dspSetExpControl
    - dspSetFlashLightControl
    - dspUpdateAutoExposure
    - dspUpdateExposure

  - dspSetEV
  - dspGetFrameYAvg
  - dspGetAECstats_sw
- Edge Enhancement Control
  - dspSetEdgeGain
- Color Correction Matrix
  - dspSetColorMtx
- Gamma Control
  - dspSetGammaTables
  - dspGetGammaTables
- Histogram Reports
  - dspSetHistogramCtrl
  - dspGetHistogramStats
- Auto Focus Control
  - dspSetAFcontrol
  - dspGetAFstats
- Bad Pixel Compensation
  - dspSetBadPixelTables
  - dspGetBadPixelTables
- Lens Shading Compensation
  - dspFindLensShadingParam
    - This function will be supported in Windows AP
  - dspLensCorrection
- Video Quality Adjustment
  - dspSetBrightnessContrast
  - dspSetHueSaturation
- Noise Reduction Filter
  - dspNoiseReduction

Used structures in dsplib.a

The types of YUV formats

```
typedef enum YUV_Data_Format
{       YUYV=0,
        YVYU,
        UYVY,
        VYUY
} YUV_DATA_FORMAT_E;
```

The types of RGB bayer format

```
typedef enum RGB_bayer_Format
{       GBRG=0,
        GRBG,
        BGGR,
        RGGB
} RGB_BAYER_FORMAT_E;
```

The definitions of Sensor output formats

```
typedef union Sensor_Output_Format
{       YUV_DATA_FORMAT_E          eYUVformat;
        RGB_BAYER_FORMAT_E         eBayerFormat;
} SENSOR_OUTPUT_FORMAT;
```

The definition of Sensor interface control

```
typedef struct Sensor_Interface
{       INT                        nSensorOutputType;
        SENSOR_OUTPUT_FORMAT       SensorOutputFormat;
        INT                        nSensorInterfMode;
        INT                        nPCLK_P;
        INT                        nHsync_P;
        INT                        nVsync_P;
        INT                        nYUV_input_type;
} SENSOR_INTERFACE_T;
```

The Settings of timing generators for Master mode sensors

```
typedef struct Frame_Timing_Gen
{       INT          nEnd_Hsync;
        INT          nTotal_Hsyncs;
        INT          nStart_PCLK;
        INT          nEnd_PCLK;
        INT          nTotal_PCLKs;
} FRAME_TIMING_GEN_T;
```

The definitions of cropping windows

```
typedef struct Crop_Start_Addr
{       UINT32      uStartX;
        UINT32      uStartY;
        UINT32      uCropWidth;
        UINT32      uCropHeight;
}       CROP_START_ADDR_T;
```

The definitions of subwindows

```
typedef struct SubWindow_Ctrl
{       INT         nStartX;
        INT         nStartY;
        INT         nSW_Width;
        INT         nSW_Height;
        INT         nSW_Xoff;
        INT         nSW_Yoff;
        INT         nAECSrc;
        INT         nAWBSrc;
} SUBWINDOW_CTRL_T;
```

The structures of digital programmable multiplier gains

```
typedef struct DPGM
{       INT         nRgain;
        INT         nGrgain;
        INT         nGbgain;
        INT         nBgain;
}       DPGM_T;
```

The definitions of white object detections

```
typedef struct White_Objects
{       BOOL        bIsSkipWhitePoint;
        UINT8       ucAWBSrc;
        BOOL        bIsDetectWO;
        UINT8       ucWO_coord;
        INT         nWO_PA;
        INT         nWO_PB;
        INT         nWO_PC;
        INT         nWO_PD;
        INT         nWO_PE;
        INT         nWO_PF;
        INT         nWO_Ka;
        INT         nWO_Kb;
        INT         nWO_Kc;
        INT         nWO_Kd;
}       WHITE_OBJECTS_T;
```

The structure of lens shading compensation

```
typedef struct Shading_Compensate_Coefficients
{       int             nSC_up;
        int             nSC_down;
        int             nSC_left;
        int             nSC_right;
} SHADING_COMP_COEFF_T;

typedef struct Shading_Comp_Control
{       INT             nSC_Shift;
        RGB_BAYER_FORMAT_E eBayerFormat;
        INT             nCenterX;
        INT             nCenterY;
        SHADING_COMP_COEFF_T *tYRcoeff;
        SHADING_COMP_COEFF_T *tUGcoeff;
        SHADING_COMP_COEFF_T *tVBcoeff;
}  SHADING_COMP_CTRL_T;
```

Sensor DSP Library API

dspEnableDSPinterrupt

- To enable Interrupt of Sensor DSP engine or not
- void dspEnableDSPinterrupt (BOOL bEnableINT)
    - Parameters :
        - bEnableINT :
          (1) TRUE : Enable the interrupt issued by the sensor DSP engine
          (2) FALSE : Disable the interrupt
    - Return value :
        - None

dspSetIRQHandler

- The callback subroutine for the application using when an interrupt occurs
- void dspSetIRQHandler (PVOID pvDspFuncPtr)
    - Parameters :
        - dspFuncPtr : The callback function pointer
    - Return :
        - None

dspGetInterruptStatus

- To retrive the status of the DSP interrupt
- UINT32 dspGetInerruptStatus (void)
    - Parameters :
        - None
    - Return :
        - bit 1 : Sensor DSP interrupt signal's mask
          (1) 0 : Disable the generation of the interrupt signal
          (2) 1 : Enable the generation of the interrupt signal
        - bit 0 : Sensor DSP interrupt status
          (1) 0 : No interrupt is generated
          (2) 1 : An interrupt is generated

dspResetDSPengine

- To reset Sensor DSP engine, but all registers' setting will be kept
- void dspResetDSPengine (void)
    - Parameters :
        - None
    - Return :
        - None

dspInitialization

- To enable DSP engine, set the sensor clock and initialize the sensor and related registers of W99702 Sensor DSP engine
- UINT64 dspInitialization (UINT32 uFrameRateCtrl, char *pcSensorType)
  - Paramters :
    - uFrameRateCtrl :
      (1) 0 : 30 fps
      (2) 1 : 20 fps
      (3) 2 : 15 fps
      (4) 3 : 10 fps
    - pcSensorType :
      Which sensor is supported in this library. From the last 2 characters, you can also know the sensor type – YUV sensors or RGB bayer sensors.
      The last 2 characters :
      - 00 : RGB bayer sensors
      - 01 : YUV422 data from YUV sensors or TV-decoder

      (1) "OV262000" ➔ OV2620 RGB mode
      (2) "OV964000" ➔ OV9640 RGB mode
      (3) "OV964001" ➔ OV9640 YUV mode
      (4) "OV764x00" ➔ OV764x series RGB mode
      (5) "OV764x01" ➔ OV764x series YUV mode
      (6) "ICM110U00" ➔ IC-Media 110U RGB mode
  - Return :
    - None

dspEnableDSPfunc

- To enable DSP functions according to users' definitions
- void dspEnableDSPfunc (UINT32 uDspFuncCtrl)
  - Parameters :
    - dspFuncCtrl :
      (1) 0 for Disable and 1 for Enable
      (2) bit 14 : Digital programmable gain multiplier
      (3) bit 13 : High saturation suppression
      (4) bit 12 : Sub-Window Statistics
      (5) bit 11 : Histogram Statistics
      (6) bit 10-9 : Bad pixel Compensation
        a. 00 : disable bad-pixel compensation
        b. 01 : μC can write bad pixel registers, but the compensation is disabled **(Reserved)**
        c. 10 : μC can read bad pixel registers, but the compensation is disabled **(Reserved)**
        d. 11 : Enable bad-pixel compensation

      (7) bit 8 : Black level clamping
      (8) bit 7 : Auto Focus
      (9) bit 6 : Peak valley filter
      (10)bit 5 : Missing color generation (MCG)
      (11)bit 4 : Color correction
      (12)bit 3 : Gamma correction
      (13)bit 2 : Color space conversion
      (14)bit 1 : Ege enhancement
  - Return :
    - None

dspDisableAllDSPfunc

- To disable all DSP functions. If all DSP functions are disabled, then the sensors' RGB 10-bit raw data can be retrived from the Y and U buffers of the planar pipe through dspGetBayerRawData (…).
- void dspDisableAllDSPfunc (void)
    - Parameters :
        - None
    - Return :
        - None

dspGetDSPfuncStatus

- To get the status of all related DSP functions
- UINT32 dspGetDSPfuncStatus (void)
    - Parameters :
        - None
    - Return :
        - The bit representations are same as dspEnableDSPfunc ()

dspSetSensorInterface

- To set sensors' output format
- void dspSetSensorInterface (Sensor_Interface *tSensorInterf)
    - Parameters :
        - Ref. to the structure definition -- **Sensor_Data_Format**
        - SensorOutputType and SensorOutputFormat :

          (1) nSensorOutputType and SensorOutputFormat :

            a. nSensorOutputType = 0 ➔ RGB bayer format data from RGB sensors
               There are 4 RGB bayer formats. Ref. to **SENSOR_OUTPUT_FORMAT** and **RGB_BAYER_FORMAT_E**

            b. nSensorOutputType = 1 ➔ YUV data from YUV sensors
               There are 4 YUV packet formats. Ref. to **SENSOR_OUTPUT_FORMAT** and **YUV_DATA_FORMAT_E**

          (2) nSensorInterfMode : The interface between W99702 DSP engine and the sensors

            a. 0 : Slave Mode
            b. 1 : Master Mode.

          (3) nPCLK_P : the polarity of PCLK latch signal

            a. 0 : Data will be latched by the negative edge
            b. 1 : Data will be latched by the positive edge

          (4) nHsync_P : The polarity of Hoziontal sync.

            a. 0 : Singal is low during Hsync. period (i.e. When signal is at low level is active sync. period)
            b. 1 : latched by the positive edge (i.e. When signal is at high level is active sync. period)

          (5) nVsync_P : The polarity of Vertical sync.
              Definitions are same as Hsync_P

          (6) nYUV_input_type

            a. 0 : Full scalar input. That is, the ranges of Y, Cb, and Cr are all 0 ~ 255
            b. 1 : Standard CCIR601. That is, the range of Y is 16 ~ 235 and the ranges of Cb and Cr are 16 ~ 240.
    - Return :
        - None

dspGetBayerRawData

- To save 10-bit RGB bayer raw data if nSensorOutputType=0 of dspSetSensorInterface (…) and all DSP functions are disabled.
- void dspGetBayerRawData (UINT32 uRawBufAddr)
    - Parameters :
        - uRawBufAddr : The starting address to store raw data. The buffer size of uRawBufAddr should be reserved at least (cropping width x cropping height x 2) bytes.
    - Return :
        - None

dspSetCroppingWnd

- To set the starting addres (x, y) and the size of the specified window
- void dspSetCroppingWnd (CROP_START_ADDR *tCropWnd)
    - Parameters :
        - Ref. to the definition of the cropping windows - **CROP_START_ADDR**.
        - uStartX : The starting address in X-axis. (Column)
        - uStartY : The starting address in Y-axis. (Line)
        - uCropWidth : The width of the image
        - uCropHeight : The height of the image
    - Return :
        - None
    - If VGA image size is captured, then uCropWidth=640 and uCropHeight=480 will be filled in this function.

dspSetSubsampling

- To set subsample mode for changing sensor output image size
- void dspSetSubsampling (INT32 nSubsmplMode)
  - Parameters :
    - nSubsmplMode : The subsample settings. **It will depend on what kind of the sensor is used and what subsample modes that sensor will provide**. And it will be noted in the dsplib.h of the released library according to different sensors. For examples,

      (1) OV2620 UXGA image sensor,
        a. nSubsmplMode = 0 : UXGA (1600x1200)
        b. nSubsmplMode = 1 : CIF (352x288)
        c. nSubsmplMode = 2 : SVGA (800x600)

      (2) OV9640 SXGA image sensor,
        a. nSubsmplMode = 0 : 1280x960 (Normal)
        b. nSubsmplMode = 1 : QCIF (176x144)
        c. nSubsmplMode = 2 : QVGA (320x240)
        d. nSubsmplMode = 4 : CIF (352x288)
        e. nSubsmplMode = 8 : VGA (640x480)

      (3) OV7640 VGA image sensors,
        a. nSubsmplMode = 0 : VGA (640x480)
        b. nSubsmplMode = 1 : QVGA (320x240)

      (4) Micron MT9M111 YUV 1.3M image sensors,
        a. nSubsmplMode = 0 : SXGA (1280x960, Normal, default mode)
        b. nSubsmplMode = 1 : VGA (640x480)
        c. nSubsmplMode = 2 : QVGA (320x240)
        d. nSubsmplMode = 3 : QQVGA (160x120)
  - Return :
    - None

dspSetTimingControl

- If a sensor is designed as the slave mode interface, this function needs to be enabled and W99702 Sensor DSP engine will produce Vsync, Hsync, and PCLKs to sensors.
- void dspSetTimingControl (FRAME_TIMING_GEN_T *tTG_Control)
    - Parameters :
        - Ref. to the definition of the structure – **FRAME_TIMING_GEN_T**
        - nEnd_Hsync : Each active Vsync period is always from the 0th line to the “end_Hsync” th line
        - nTotal_Hsyncs : The total lines (Hsync.) per frame (Including the whole sensing area)
        - nStart_PCLK and nEnd_PCLK : Each active Hsync period is always from the “start_PCLK” th pixel to “end_PCLK” th pixel
        - nTotal_PCLKs : The total pixel (PCLK) per Hsync.
    - Return :
        - None

dspSetSubWindow

- To set the subwindows for AEC and AWB to calculate statistics for each subwindow
- void dspSetSubWindow (SUBWINDOW_CTRL_T *tSubWndCtrl)
    - Parameters :
        - The subwindow control can be referenced to the definition of the structure – **SUBWINDOW_CTRL_T**
        - nStartX : The starting address in X-axis for the subwindow (0, 0)
        - nStartY : The starting address in Y-axis for the subwindow (0, 0)
        - nSW_Width : The width of each subwindow
        - nSW_Height : The height of each subwindow
        - nSW_Xoff : The offset between two adjacent subwindows in X-axis
        - nSW_Yoff : The offset between two adjacent subwindows in Y-axis
        - nAECSrc : The source of the AEC statistics for each subwindow will be calculated before DPGM or after DPGM
            - nAECSrc = 0 : Before DPGM
            - nAECSrc = 1 : After DPGM
        - nAWBSrc : The source of the AWB statistics for each subwindow will be calculated before DPGM or after DPGM
            - nAECSrc = 0 : Before DPGM
            - nAECSrc = 1 : After DPGM
    - Return :
        - None

dspSetBLCwnd

- To define a sub window for black level compensation using
- void dspSetBLCwnd (CROP_START_ADDR_T *tBlcWnd)
  - Parameters :
    - The defintion of the structure –CROP_START_ADDR_T
    - uStartX : The starting address in X-axis
    - uStartY : The starting address in Y-axis
    - uCropWidth / uCropHeight : The width and height of the window for black level. **The usages of uCropWidth and uCropHeight are different from the usages in dspSetCroppingWnd(…). It is just an index in this functions.**
      (1) 0 : 8
      (2) 1 : 16
      (3) 2 : 32
      (4) 3 : 64
      (5) 4 : 128
      (6) 5 : 256
      (7) 6 : 512
      (8) 7 : 1024
  - Return :
    - None

dspSetBLCMode

- To set the black level clamping mode
- void dspSetBLCMode (BOOL bIsUserDefined, BOOL bIsAutoBLC)
  - Parameters :
    - bIsUserDefined :
      (1) 0 : disable user-defined mode
      (2) 1 : enable user-defined mode
    - bIsAutoBLC :
      (1) 0 : disable auto-detection black level clamping
      (2) 1 : enable auto-detection black level clamping
  - Return :
    - None

dspUpdateUserBlackLevels

- To set the user-defined black level for each color filter. This function will be valid only when bUserDefined=1 in dspSetBLCMode ()
- void dspUpdateUserBlackLevels (INT nUDBL_Gr, INT nUDBL_Gb, INT nUDBL_R, INT nUDBL_B)
  - Parameters :
    - nUDBL_Gr : The user-defined black level for Gr channel.
    - nUDBL_Gb : The user-defined black level for Gb channel.
    - nUDBL_R : The user-defined black level for R channel.
    - nUDBL_B : The user-defined black level for B channel.
    - The limitation for these four variables is between -128 and 127.
  - Return :
    - None

dspGetDetectedBlackLevel

- To get the average black level from the user-defined subwindow – defined in dspSetBLCMode () with bAutoBLC=TRUE
- INT32 dspGetDetectedBlackLevel (void)
  - Parameters :
    - None
  - Return :
    - The detected black level in auto-detected mode

dspFalseColorSupp

- To do false color suppression to eliminate some false color
- void dspFaseColorSupp (INT nFCS_factor)
  - Parameters :
    - nFCS_factor :
      (1) 0 : 1.0x. That is, there is no effect of the false color suppression
      (2) 2 : 0.5x. That is, if this point (R, G, B) is regarded as false color, its color will be equal to the half of the original color.
      (3) 3 : 0.25x. That is, if this point (R, G, B) is regarded as false color, its color will be equal to one-fourth of the original color.
  - Return :
    - None

dspSetSceneMode

- To set different scenes for AEC and AWB using
- void dspSetSceneMode (INT32 nSceneMode)
  - Parameters :
    - nSceneMode : ***(These definitions are temporary)***
      (1) 0 : Normal condition 1
      (2) 1 : Normal condition 2
      (3) 2 : Portrait
      (4) 3 : Indoor 60Hz
      (5) 4 : Indoor 50Hz
      (6) 5 : Night mode
      (7) 6 : User-defined mode
  - Return :
    - None

dspSetAWBcontrol

- To set the relative AWB controls for white object detection using.
- void dspSetAWBcontrol (CROP_START_ADDR_T *tCropWO, WHITE_OBJECTS_T *tDefWO)
  - Parameters :
    - tCropWO : Ref. to the defintion of the cropping window -- **CROP_START_ADDR_T**
    - tDefWO : Ref. to the definition of the structure -- **WHITE_OBJECTS_T**
      - bIsSkipWhitePoint :
        (1) FALSE : If at least one of R, G or B value is saturated, it still calculate these points.
        (2) TRUE : If at least one of R, G or B value is saturated, then this pixel will not be calculated.
      - ucAWBSrc : The source data used for AWB calculation
        (1) 0 : The statistical data will be calculated before DPGM
        (2) 1 : The statistical data will be calculated after DPGM
      - bIsDetectedWO :
        (1) FALSE : The detection of white objects will not be done. That is, all pixels which exist in defined white object window will be calculated.
        (2) TRUE : The pixels which only satisfy the definition of white objects will be calculated.
      - ucWO_coord :
        (1) 0 : B-Y and R-Y (i.e. Use Y as a referenced axis)
        (2) 1 : B-G and R-G (i.e. Use G as a referenced axis)
      - nWO_PA, nWO_PB, nWO_PC, nWO_PD, nWO_PE, nWO_PF, nWO_Ka, nWO_Kb, nWO_Kc, nWO_Kd : The definitions of white objects for a specified sensor.
  - Return :
    - None

dspUpdateAutoWhiteBalance

- To do color balance automatically.
- void dspUpdateAutoWhiteBalance (void)
  - Parameters :
    - None
  - Return :
    - None

dspUpdateWhiteBalance

- To do adjust color balance manually
- void dspUpdateWhiteBalance (DPGM_T *tColorBalance)
  - Parameters :
    - The definition of the structure – **DPGM_T**
    - nRgain : The digital gain is applied on R channel
    - nGrgain : The digital gain is applied on Gr channel
    - nGbgain : The digital gain is applied on Gb channel
    - nBgain : The digital gain is applied on B channel
  - Return :
    - None

dspGetColorGains

- To get the 4 color digital gains which are applied in Sensor DSP engine now.
- void dspGetColorGains (DPGM_T *tColorBalance)
  - Parameters :
    - Ref. to the definition of the strucutre – **DPGM_T** and the description in dspUpdateWhiteBalance (…)
  - Return :
    - None

dspGetAWBstats

- To report the R, G or B average of all pixels defined in the white object window.
- UINT32 dspGetAWBstats (void)
  - Parameters :
    - None
  - Return :
    - bit [31:24] : Reserved
    - bit [23:16] : The R average of pixels defined in the white object window
    - bit [15:8] : The G average of pixels defined in the white object window
    - bit [7:0] : The B average of pixels defined in the white object window

dspGetAWBstats_sw

- To caculate the average on R channel, G channel or B channel of each subwindow
- void dspGetAWBstats_sw (UINT8 *pucAvgR, UINT8 *pucAvgG, UINT8 *pucAvgB)
  - Parameters :
    - The length of the average of each subwindow is 8-bit long and the total length of 16 subwindows for average R, average G or average B for 16 subwindows should be 8x16 = 128 bytes
    - pucAvgR : The pointer of the buffer for R averages of 16 subwindows
    - pucAvgG : The pointer of the buffer for G averages of 16 subwindows
    - pucAvgB : The pointer of the buffer for B averages of 16 subwindows
  - Return :
    - None

dspGetWOcounts

- To report the number of pixels which will satisfy the definition of white objects.
- UINT32 dspGetWOcounts (void)
  - Parameters :
    - None
  - Return :
    - bit[15:0] = The number of the white object points. (= total_WO_Counts / 32)

dspSetExpControl

- To set the target luminous.
- void dspSetExpControl (int nAE_targetLum, int nForeGndRatio, CROP_START_ADDR_T *tForeWin, UINT8 ucAECsrc)
  - Parameters :
    - nAE_targetLum : To set the target luminance for auto exposure using.
    - nForeGndRatio : To set the ratio of the foreground window and this value is valid from 0 ~ 16. And (16-ForeGndRatio) is the ratio of the background.
    - tForeWin : Ref. to the structure **CROP_START_ADDR_T** and **dspSetCroppingWnd (…)**
    - ucAECsrc : The source data used for AEC caculation.
      (1) 0 : Before DPGM (The data will be caculated before DPGM is applied.)
      (2) 1 : After DPGM (The data will be caculated after DPGM is applied.)
  - Return :
    - None

dspSetFlashLightControl

- To set the flash light control
- void dspSetFlashLightControl (UINT32 uFlashLightMode)
  - Parameters :
    - uFlashLightMode : the control of the flash light
      (1) 0 : Automatically. It will depend on the report from dspUpdateAutoExposure (…) or dspUpdateExposure (…)
      (2) 1 : The flash light will be enforced to be turned on
      (3) 2 : The flash light will not be turned on.
  - Return :
    - None

dspUpdateAutoExposure

- To do auto exposure control
- UINT32 dspUpdateAutoExposure (void)
  - Parameters :
    - None
  - Return :
    - bit 0 : The report of AEC status
      (1) 0 : AEC is still in the unstable status
      (2) 1 : AEC is in the stable status
    - bit 1 : The status of the flash light control
      (1) 0 : The flash light do not need to be turned on
      (2) 1 : The flash light should be turned on

dspUpdateExposure

- To update exposure control manually
- void dspUpdateExposure (UINT32 uExpTime, UINT32 uAGC)
  - Parameters :
    - uExpTime : To control sensors's exposure time
    - uAGC : To set auto gain control
  - Return :
    - bit 0 : The report of AEC status
      (1) 0 : AEC still in the unstable status
      (2) 1 : AEC in the stable status
    - bit 1 : The status of the flash light control
      (1) 0 : The flash light do not need to be turned on
      (2) 1 : The flash light should be turned on

dspSetEV

- To update EV control
- void dspSetEV (int nEV)
  - Parameters :
    - nEV : The EV value
  - Return :
    - None

dspGetFrameYAvg

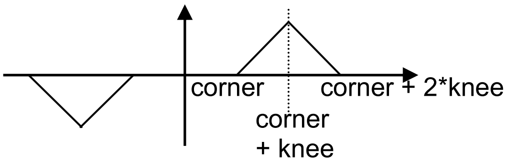
- To get the Y averages of the foreground and whole window
- UINT32 dspGetFrameYAvg (void)
  - Parameters :
    - None
  - Return :
    - bit [31:16] : Reserved
    - bit [15:8] : The average luminance of the whole window
    - bit [7:0] : The average luminance of the foreground window

dspGetAECstats_sw

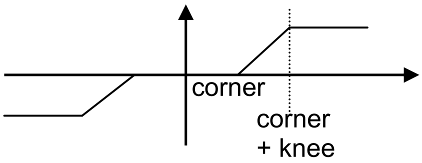
- To get the Y averages of 16 subwindows
- void dspGetAECstats_sw (UINT8 *pucAvgY)
  - Parameters :
    - pucAvg : The Y averages of 16 subwindows. The length of each subwindow for average Y is 8-bit long.
  - Return :
    - None

dspSetEdgeGain

- To set the the strength of the edge enhancement
- void dspSetEdgeGain (int nKneeMode, int nKneePoint, int nCornerPoint, int nEdgeGain)
  - Parameters :
    - nKneeMode : Which kind of the edge enhancement will be done
      (1) 0 : The enhanced strength after the corner point will be decreased



      (2) 1 : The enhanced strength after the corner point will be the same as the corner point



    - nKneePoint : Edge knee point
    - nCornerPoint : Edge corner point
    - nEdgeGain : The strength of the edge enhancement (0 ~ 31 ➔ 0.0x ~ 7.75x)
  - Return :
    - None

dspHSS

- To enable high saturation suppression to eliminate some over-saturated color.
- void dspHSS (int nHSS_Fa2, int nHSS_Fa1, int nHSS_Point)
  - Parameters :
    - nHSS_Fa2 : Tuning of HSS_Fa1 (Smaller HSS_Fa2 results in more saturated colors)
    - nHSS_Fa1 : HSS_Fa1 – 255/HSS_Point
    - nHSS_Point : Hight saturation suppression will be active from the gray level = HSS_Point
  - Return
    - None

dspSetColorMtx

- To set the color correction matrix
- void dspSetColorMtx (INT32 nCCMtx[3][3])
  - Parameters :
    - nCCMtx : The 3x3 color corrected matrix. The representation is shown as following :

$$\begin{bmatrix} CCMtx[0][0] & CCMtx[0][1] & CCMtx[0][2] \\ CCMtx[1][0] & CCMtx[1][1] & CCMtx[1][2] \\ CCMtx[2][0] & CCMtx[2][1] & CCMtx[2][2] \end{bmatrix}$$

The relations of Color Correction matrix :

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} CCMtx[0][0] & CCMtx[0][1] & CCMtx[0][2] \\ CCMtx[1][0] & CCMtx[1][1] & CCMtx[1][2] \\ CCMtx[2][0] & CCMtx[2][1] & CCMtx[2][2] \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

  - Return :
    - None

dspSetGammaTables

- To set the gamma correction type and 16/32 gamma entries
- void dspSetGammaTables (UINT32 nGamType, UINT16 usGamTbl[64])
  - Parameters :
    - nGamType : The gamma correction type
      (1) 0 : Table mapping for R, G, B. Same W99682's gamma correction
      (2) 1 : Reserved
      (3) 2 : Multiplication for R, G, B. And use Y as the referenced point.
      (4) 3 : Multiplication for R, G, B. And use ((Y+MAX(R,G,B)/2) as the referenced point
    - usGamTbl
      (1) if gamType = 0, there are 64 gamma entries used in the gamma correction. Each gamma entry is 8-bit unsigned integer.
      (2) if gamType=2 or 3, there are only 32 gamma entries used in the gamma correction. Each gamma entry (multiplication factor) consists of 3-bit integer and 7-bit decimal fraction.
  - Return :
    - None

dspGetGammaTables

- To get the gamma correction type and 32/64 gamma entries
- UINT32 GetGammaTables (UINT32 *puGamTbl)
  - Parameters :
    - puGamTbl :
      (1) 32 entries for gamma correction type is set 2 or 3. That is, the length of the GamTbl should be (16*32) bytes
      (2) 64 entries for gamma correction type is set 0. That is, the length of each gamma entry is 8-bit long and the length of GamTbl is = (16*32) bytes
  - Return :
    - The returned value is the gamma type. The definition is referenced to the dspSetGammsTables ()

dspSetHistogramCtrl

- To enable the gray level statistics of the image and control its related settings
- void dspSetHistogramCtrl (int nHistoSrc, int nHistoScalar, int nHistoSrcChannel)
    - Parameters :
        - nHistoSrc :
            1. 0 : Before DPGM
            2. 1 : After DPGM
        - nHistoScalar :
            1. 0 : the factor of reported values = 2
            2. 1 : the factor of reported values = 4
        - nHistoSrcChannel :
            1. 0 : R channel
            2. 1 : G channel
            3. 2 : B channel
            4. 3 : Y=(5R+9G+2B)/16
            5. 4 : MAX (R, G, B)
    - Return :
        - None

dspGetHistogramStats

- To get the statistics of 12-step histogram
- void dspGetHistoStats (UINT16 *pucHistoPtr)
    - Parameters :
        - pucHistoPtr : The 12-step histogram. The length of pHisto should be at least (12*16) bytes.
        - Each step is 16-bit long.
    - Return :
        - None

dspSetAFcontrol

- Set Target value and the effective window
- void dspSetAFcontrol (CROP_START_ADDR_T *tAFcontrol)
    - Parameters :
        - tAFcontrol : The definition of the structure – CROP_START_ADDR_T
    - Return :
        - None

dspGetAFstats

- To return the statistics of the high-pass frequence domain in the specified window
- UINT32 dspGetAFreports (void)
    - Parameters :
        - None
    - Return value :
        - The format of 32-bit return values : the LSB 24 bits (bit 23 ~ bit 0) of the high-pass frequence

dspSetBadPixelTables

- To set the bad pixels and the maximum number of bad pixels is 32.
- void dspSetBadPixelTables (int nBadPixCnt, UINT32 *puBPtblAddr)
  - Parameters :
    - nBadPixCnt : The total number of bad pixels and BadPixCnt should be at most 32.
    - puBPtblAddr : The tables for bad pixels. And its address format (x, y) is :
      (1) bit [27:16] : X position of a bad pixel
      (2) bit [11:0] : Y position of a bad pixel
  - Return :
    - None

dspGetBadPixelTables

- To retrieve the number of the bad pixel and its position.
- void dspGetBadPixelTables (UINT32 *puBPtblAddr)
  - Parameters :
    - puBPtblAddr : The address of the buffer for storing bad pixels' table
      (1) The 1st word : the number of the bad pixels.
      (2) The words after the 1st word are positions of bad pixels. And one word representation (x, y) :
        - bit [27:16] : the X position of a bad pixel
        - bit[11:0] : the Y position of a bad pixel
  - Return :
    - None

dspLensCorrection

- To compensate the lens shading
- void dspLensCorrection (SHADING_COMP_CTRL_T *tSCctrl)
  - Parameters :
    - tSCctrl : Ref. to the definition of the structure -- **SHADING_COMP_CTRL_T**
      (1) nSC_shift : The shift value for all coefficients
        a. 0 : SC_Shift = 17
        b. 1 : SC_Shift = 18
        c. 2 : SC_Shift = 19
        d. 3 : SC_Shift = 20
      (2) eBayerFormat : The color filter array (bayer format). And this setting will be a little different from the color filter array in dspSetSensorInterface (…). Ref. to the defined emulator – **RGB_BAYER_FORMAT_E**
      (3) nCenterX : The address of the referenced center point in X-axis
      (4) nCenterY : The address of the referenced center point in Y-axis
      (5) tYRcoeff, tUGcoeff, tVBcoeff : The compensated parabola coefficients for YUV output format or RGB output format individually. Ref. to the definition of the structure – **SHADING_COMP_COEFF_T**
        a. nSC_up : Parabola coefficient at the position < Y0
        b. nSC_down : Parabola coefficient at the position > Y0
        c. nSC_left : Parabola coefficient at the position < X0
        d. nSC_right : Parabola coefficient at the position > X0
  - Return :
    - None

dspFindLensCorrectionParam

- To dump a raw data picture or YUV data (packet or planar) and let AP on PC find related parameters of the lens shading compensation.

dspSetBrightnessContrast

- To make adjustments to the tonal range of an image
- void dspSetBrightnessContrast (int nYscale, int nYoffset)
    - Parameters :
        - nYscale : The contrast adjustment. Its range is 0 ~ 63 and the default value is 16.
        - nYoffset : The adjustment of the brightness. Range : -128 ~ 127
        - Y′ = Y x nYoffset + nYoffset
    - Return :
        - None

dspSetHueSaturation

- To adjust the hue and saturation of the entire image
- void dspSetHueSaturation (int nHS1, in nHS2)
    - Parameters :
        - nHS1 and nHS2:
          (1) nHS1 will influence the saturation. Its range is 127 ~ -128 and the default value is 32.
          (2) nHS2 will influence the hue. Its range is 127 ~ -128 and the default value is 0.
          (3) Cb′ = Cb x nHS1 – Cr x nHS2
          (4) Cr′ = Cb x nHS2 – Cr x nHS1
    - Return :
        - None

dspNoiseReduction

- To reduce the noise reduction or not
- void dspNoiseReduction (BOOL bEnableNR)
    - Parameters :
        - bEnableNR :
          (1) TURE : Enable the noise reduction
          (2) FALSE : Disable the noise reduction
    - Return :
        - None

Sample Code

- ***For General Users***
  - **For RGB sensors,**

```
extern void dspInitHandler ();

dspResetDSPengine ();
dspInitialization (0);
dspSetIRQHandler ((PVOID)dspInitHandler);
dspEnableDSPinterrupt (TRUE);

while (1)
{       dspUpdateAutoExposure ();
        dspUpdateAutoWhiteBalance ();
}
```

  - **For YUV sensors,**

```
dspInitialization (0);      //Only this function is needed
```

- ***For Advanced users,***
  - **For RGB sensors,**

```
extern void dspInitHandler ();

dspResetDSPengine ();
dspInitialization ();
dspSetIRQHandler ((PVOID)dspInitHandler);
dspEnableDSPinterrupt ();

//You can control related DSP functions and settings anytime
//e.g. dspSetEdgeGain ()      -- to set different
//            dspSetGammaTables () – you can specify it
//            dspUpdateUserBlackLevels ()
//            …………………….

while (1)
{       dspUpdateExposure ();
        dspUpdateWhiteBalance ();
}
```

- **For YUV sensors,**

```
int nConstrast = 10;      //For example
int nBrightness = 20;     //For example
int nHue = 0x20;          //For example
int nSaturation = 0x20;   //For example

dspInitialization (0);

while (1)
{       dspSetBrightnessContrast (nConstrast, nBrightness);
        dspSetHueSaturation (nHue, nSaturation);
        dspLensCorrection (…);       //Should depend the lens shading compensation parameters
}
```

Special Programming Notes.

- Bad Pixels :
  - When writing bad pixels’ address (x, y) and index into DSP control registers, it should use HCLK to write.
  - When reading bad pixels’ address (x, y) and index into DSP control registers, it should use ECLK to read.
  - Bad pixels’ real position should be (x+3, y+2) as compared with captured image after cropping.
- Cache Issue,
  - With cache on,
    - Fast Serial bus (H/W I2C) : Prefer to do I2C initialization before WB_EnableCache (CACHE_WRITE_THOUGH) or WB_EnableCache(CAHCH_WRITE_BACK)
    - Software I2C : The delay between some signals should be longer
- For slave mode sensors, the polarities of PCLK, Hsync., and Vsync. are all defined in Video capture engine.
- ***All related DSP initializations should be called before the initialization of video capture engine.***
  - For example, dspInitialization (0, SensorMdl);
-