Dell SDK for Monitors

Application Programming Interface Guide

for SDK version 3.0

2019 – 09 Rev. A00

Contents

Introduction

This document describes the APIs for supported Dell UltraSharp monitors on Linux(x86), OSX and Windows platforms. These APIs are to be used for remote display management and control from a Host PC to supported Dell UltraSharp monitors via a USB connection. A USB 3.0 A to B cable should be used for the connection between the host and the display. For UP2720Q, connecting to the monitor can either be USB A to Thunderbolt cable, or Thunderbolt to Thunderbolt cable.

The following monitors are supported:

1. UP2516D
2. UP2716D
3. UP3017
4. UP3218K
5. UP2718Q
6. UP2720Q

The API described in this document corresponds to SDK version 3.0. Please refer to the SDK compliance checklist of your model for information on possible deviations with some APIs.

API Return Codes

All APIs return a MONITOR_CODE as described below:

Return

MONITOR_CODE	Code describing the result of the API call
	0 Success
	1 Timeout
	2 Parameters Error
	3 Connection error with monitor
	4 Communications error with monitor
	5 Wrong state for API call
	6 API not supported by monitor
	7 Checksum error
	8 Error due to related module powered off
	-1 Other Failure

```
typedef enum monitor_code
{
    MONITOR_SUCCESS       = 0,
    MONITOR_FAILURE       = -1,
    MONITOR_ERR_TIMEOUT   = 1,
    MONITOR_ERR_PARAMS    = 2,
    MONITOR_ERR_CONNECT   = 3,
    MONITOR_ERR_COMMS     = 4,
    MONITOR_ERR_STATE     = 5,
    MONITOR_ERR_NOSUPPORT = 6,
    MONITOR_ERR_CHECKSUM  = 7,
    MONITOR_ERR_MODULEOFF = 8,
    MONITOR_ERR_INUSE     = 9
}
MONITOR_CODE;
```

Error Codes Explanation

MONITOR_ERR_TIMEOUT	Returned when user did not respond to the SDK acknowledgement OSD prompt after 30s
MONITOR_ERR_PARAMS	Called the API with invalid, out of range values. For example, sending a value of 200 for SetSharpness
MONITOR_ERR_CONNECT	1. No available or compatible monitors detected to connect to. 2. Error in opening the HID monitor device for communications. 3. Error in opening the HID monitor device's MCU for communications.
MONITOR_ERR_COMMS	**Fatal** communications error where communications broke down between SDK and the monitor. All further commands from this point onwards will likely result in the same error. Unrecoverable via software, may need to power cycle monitor. Suitable message in English would be: *"Communication with monitor failed. Please close this application and restart."*
MONITOR_ERR_STATE	API cannot be called in the current monitor state. For example, some monitors will need to be in Color Preset > Color Space before being able to SetColorSpaceState
MONITOR_ERR_NOSUPPORT	Calling an API for a monitor without the functionality. For example, calling SetPxPLocation on a UP2720Q
MONITOR_ERR_CHECKSUM	Checksum error on reading back LUT data
MONITOR_ERR_MODULEOFF	Calling an API that requires certain module to be ON. For example, calling GetCalibrationResult when the Calibration Module Power = OFF
MONITOR_ERR_INUSE	Cannot start a session with monitor as it is already communicating with another application in a different session. Application should check the returned token and display appropriate message that the monitor is currently in use by another application. Example message in English would be: *"<LABEL> software is communicating with the monitor, please quit it before launching this application again."*
MONITOR_FAILURE	1. When user rejects the session when prompted on OSD 2. Any other failure not covered above by other ERROR codes

Token	<LABEL>
0x0001	CalMAN Calibration
0x0006	Dell Color Management
0x0007	Dell SelfCal Administrator
0x000F	Dell SDK
0x0010-0xFFFE	Another

Monitor Management

Initialize

Initialize the SDK before first use

API
MONITOR_CODE Initialize(void)

Params
-

Shutdown

Shuts down the SDK at the end of use

API
MONITOR_CODE Shutdown(void)

Params
-

GetAvailableMonitors

Returns the number of supported monitor(s)

API
MONITOR_CODE GetAvailableMonitors(BYTE *pu8Val)

Params

*pu8Val	Pointer to return number of supported monitors connected

Return

pu8Val	Number of supported monitors connected

GetAvailableMonitorsDetail

Returns the number of supported monitor(s) and details like name and whether an inbuilt colorimeter is present.

API
MONITOR_CODE GetAvailableMonitorsDetail(BYTE *pu8Count, MonitorDetailStructType **arrMonitorDetail)

Params

*pu8Count	Pointer to return number of supported monitors connected
**arrMonitorDetail	Pointer to an array of supported monitor details

Return

pu8Count	Number of supported monitors connected
arrMonitorDetail[0…n]	Array length = Number of supported monitors n = Number of supported monitors - 1

```
typedef struct MonitorDetailStruct {
    BYTE MonitorName[11];
    BYTE ServiceTag[8];
    BYTE InbuiltColorimeter; // 0 = No, 1 = Yes
    BYTE ColorimeterName[11];
}
MonitorDetailStructType;
```

ConnectMonitor

Connect to monitor and start session. Acknowledge SDK Access in OSD menu must be Enabled

API
MONITOR_CODE ConnectMonitor(BYTE u8Val)

Params

u8Val	Index of monitor as returned by GetAvailableMonitors to connect to. Index starts at 0 for the first monitor.

ConnectMonitorByServiceTag

Connect to monitor and start session. Acknowledge SDK Access in OSD menu must be Enabled

API
MONITOR_CODE ConnectMonitorByServiceTag(BYTE *serviceTag)

Params

serviceTag	Service Tag of monitor as returned by GetAvailableMonitorsDetail to connect to.

DisconnectMonitor

Disconnect to monitor and end session. If cable is unplugged and KeepAlive returns an error, this command must be called before reconnection to the monitor after the cable is plugged back in.

API
MONITOR_CODE DisconnectMonitor(void)

Params
-

SetAssetTag

Set the asset tag of the monitor.

API
MONITOR_CODE SetAssetTag(BYTE *pbyAssetTag)

Params

*pbyAssetTag	Pointer to asset tag ID string (max 10 chars)

GetAssetTag

Returns the monitor asset tag. Asset Tag will be empty until set by SetAssetTag.

API
MONITOR_CODE GetAssetTag(BYTE *pbyAssetTag)

Params

*pbyAssetTag	Pointer to return asset tag ID string

Return

pbyAssetTag	Asset tag ID string (max 10 chars)

GetMonitorName

Returns the monitor name

API
MONITOR_CODE GetMonitorName(BYTE *pbyMonitorName)

Params

*pbyMonitorName	Pointer to return monitor name

Return

pbyMonitorName	Monitor name string (max 10 chars)

GetMonitorSerialNumber

Returns the monitor serial number

API
MONITOR_CODE GetMonitorSerialNumber(BYTE *pbySerialNumber)

Params

*pbySerialNumber	Pointer to return monitor serial number

Return

pbySerialNumber	Monitor serial number string (max 12 chars)

GetBacklightHours

Returns the monitor backlight hours

API
MONITOR_CODE GetBacklightHours (UWORD16 *pu16Val)

Params

*ps16Val	Pointer to return monitor backlight hours

Return

ps16Val	Monitor backlight hours

GetServiceTag

Returns the monitor service tag

API
MONITOR_CODE GetServiceTag(BYTE *pbyServiceTag)

Params

*pbyServiceTag	Pointer to return monitor service tag

Return

pbyServiceTag	Monitor service tag string (max 12 chars)

IdentifyMonitor

Identify supported monitors starting with index 1.

API
MONITOR_CODE IdentifyMonitor(void)

Params
-

Power Management

GetPowerState

Returns the current power state of the monitor

API
MONITOR_CODE GetPowerState(BYTE *pu8Val)

Params

*pu8Val	Pointer to return power state

Return

pu8Val

```
typedef enum power_state
{
    POWER_STATE_OFF     = OFF,
    POWER_STATE_ON      = ON,
    POWER_STATE_STANDBY = 2
}
POWER_STATE;
```

SetPowerState

Set the monitor on, off or standby

API
MONITOR_CODE SetPowerState(BYTE u8Val)

Params

u8Val

```
typedef enum power_state
{
    POWER_STATE_OFF     = OFF,
    POWER_STATE_ON      = ON,
    POWER_STATE_STANDBY = 2
}
POWER_STATE;
```

GetPowerLED

Returns the power LED setting of the monitor

API
MONITOR_CODE GetPowerLED(BYTE *pu8Val)

Params

*pu8Val	Pointer to return power LED setting

Return

pu8Val — Power LED Setting

0	Off during Active
1	On during Active

SetPowerLED

Set the power LED setting

API
MONITOR_CODE SetPowerLED(BYTE u8Val)

Params

u8Val	Power LED Setting 0 Off during Active 1 On during Active

GetPowerUSB

Returns the power USB setting of the monitor

API
MONITOR_CODE GetPowerUSB(BYTE *pu8Val)

Params

*pu8Val	Pointer to return power USB setting

Return

pu8Val	Power USB Setting 0 Off during Standby 1 On during Standby

```
typedef enum power_usb
{
    POWER_USB_OFF = OFF,
    POWER_USB_ON  = ON
}
POWER_USB;
```

SetPowerUSB

Set the power USB setting

API
MONITOR_CODE SetPowerUSB(BYTE u8Val)

Params

u8Val	Power USB Setting 0 Off during Standby 1 On during Standby

```
typedef enum power_usb
{
    POWER_USB_OFF = OFF,
    POWER_USB_ON  = ON
}
POWER_USB;
```

GetPowerTBT

Returns the power TBT setting of the monitor

API
MONITOR_CODE GetPowerTBT(BYTE *pu8Val)

Params

*pu8Val	Pointer to return power TBT setting

Return

pu8Val	Power USB Setting 0 Off during Standby 1 On during Standby

```
typedef enum power_tbt
{
    POWER_TBT_OFF = OFF,
    POWER_TBT_ON  = ON
}
POWER_TBT;
```

SetPowerTBT

Set the power USB setting

API
MONITOR_CODE SetPowerTBT(BYTE u8Val)

Params

u8Val	Power USB Setting 0 Off during Standby 1 On during Standby

```
typedef enum power_tbt
{
    POWER_TBT_OFF = OFF,
    POWER_TBT_ON  = ON
}
POWER_TBT;
```

Image Management

GetBrightness

Returns the brightness level of the monitor

API
MONITOR_CODE GetBrightness(BYTE *pu8Val)

Params

*pu8Val	Pointer to return brightness value

Return

pu8Val	Brightness value Integer value 0 (dark) to 100 (bright) Default 75 Values in increments of 1

SetBrightness

Set the brightness level of the monitor

API
MONITOR_CODE SetBrightness(BYTE u8Val)

Params

u8Val	Brightness value Integer value 0 (dark) to 100 (bright) Default 75 Values in increments of 1

GetContrast

Returns the contrast level of the monitor

API
MONITOR_CODE GetContrast(BYTE *pu8Val)

Params

*pu8Val	Pointer to return contrast value

Return

pu8Val	Contrast value Integer value 0 (minimal) to 100 (maximum) Default 75 Values in increments of 1

SetContrast

Set the contrast level of the monitor.

NOTE: Uniformity Compensation must be turned off for this to work.

API
MONITOR_CODE SetContrastUBYTE u8Val)

Params

u8Val	Contrast value Integer value 0 (minimal) to 100 (maximum) Default 75 Values in increments of 1

GetDynamicContrast

Returns the dynamic contrast setting. Applicable for Movies and Gaming.

NOTE: Only works in Color Preset Game or Movie.

API
MONITOR_CODE GetDynamicContrast(BYTE *pu8Val)

Params

*pu8Val	Pointer to return dynamic contrast value

Return

pu8Val	Dynamic Contrast	
	0	Off
	1	On

SetDynamicContrast

Turns on/off the dynamic contrast setting. Applicable for Movies and Gaming.

NOTE: Only works in Color Preset Game or Movie.

API
MONITOR_CODE SetDynamicContrast(BYTE u8Val)

Params

u8Val	Dynamic Contrast	
	0	Off
	1	On

GetAspectRatio

Returns the aspect ratio

API
MONITOR_CODE GetAspectRatio(BYTE *pu8Val)

Params

*pu8Val	Pointer to return aspect ratio

Return

pu8Val

```
typedef enum aspect_ratio
{
    ASPECT_RATIO_WIDE    = 0x00,
    ASPECT_RATIO_AUTO    = 0x01,
    ASPECT_RATIO_4X3     = 0x02,
    ASPECT_RATIO_1X1     = 0x03,
    ASPECT_RATIO_WIDTH   = 0x04,
    ASPECT_RATIO_HEIGHT  = 0x05,
    ASPECT_RATIO_17X9    = 0x06,
    ASPECT_RATIO_16X9    = 0x07,
    ASPECT_RATIO_PIXEL   = 0x08,
}
ASPECT_RATIO;
```

SetAspectRatio

Sets the aspect ratio

API
MONITOR_CODE SetAspectRatio(BYTE u8Val)

Params

u8Val

```
typedef enum aspect_ratio
{
    ASPECT_RATIO_WIDE    = 0x00,
    ASPECT_RATIO_AUTO    = 0x01,
    ASPECT_RATIO_4X3     = 0x02,
    ASPECT_RATIO_1X1     = 0x03,
    ASPECT_RATIO_WIDTH   = 0x04,
    ASPECT_RATIO_HEIGHT  = 0x05,
    ASPECT_RATIO_17X9    = 0x06,
    ASPECT_RATIO_16X9    = 0x07,
    ASPECT_RATIO_PIXEL   = 0x08,
}
ASPECT_RATIO;
```

GetSharpness

Returns the sharpness level

API
MONITOR_CODE GetSharpness(BYTE *pu8Val)

Params

*pu8Val	Pointer to return sharpness value

Return

pu8Val	Sharpness value Integer value 0 to 100 Default 50 Values in increments of 10

SetSharpness

Sets the sharpness level

API
MONITOR_CODE SetSharpness(BYTE u8Val)

Params

u8Val	Sharpness value Integer value 0 to 100 Default 50 Values in increments of 10

GetResponseTime

Returns the response time

API
MONITOR_CODE GetResponseTime(BYTE *pu8Val)

Params

*pu8Val	Pointer to return response time value

Return

pu8Val

Response Time

0	Normal
1	Fast

```
typedef enum response_time
{
      RESPONSE_TIME_NORMAL = 0,
      RESPONSE_TIME_FAST   = 1
}
RESPONSE_TIME;
```

SetResponseTime

Sets the response time

API
MONITOR_CODE SetResponseTime(BYTE u8Val)

Params

u8Val	Response Time 0 Normal 1 Fast

```
typedef enum response_time
{
      RESPONSE_TIME_NORMAL = 0,
      RESPONSE_TIME_FAST   = 1
}
RESPONSE_TIME;
```

GetHDR

Returns the HDR setting

API
MONITOR_CODE GetHDR(BYTE *pu8Val)

Params

*pu8Val	Pointer to return HDR setting value

Return
pu8Val

```
typedef enum hdr
{
    HDR_OFF = 0,
    HDR_NORMAL = 1,
    HDR_VIVID = 2
}
HDR;
```

SetHDR

Sets the HDR setting

API
MONITOR_CODE SetHDR(BYTE u8Val)

Params
u8Val

```
typedef enum hdr
{
    HDR_OFF = 0,
    HDR_NORMAL = 1,
    HDR_VIVID = 2
}
HDR;
```

GetDCIMasking

Returns the DCI masking setting

API
MONITOR_CODE GetDCIMasking(BYTE *pu8Mask, BYTE *pu8Opacity)

Params

*pu8Mask	Pointer to return show masked setting
*pu8Opacity	Pointer to return masked opacity setting

Return

pu8Mask	Show Masked Region 0 No 1 Yes
pu8Opacity	Mask Opacity value Integer value 0, 20, 40, 60, 80 or 100

SetDCIMasking

Sets the DCI masking setting

API
MONITOR_CODE SetDCIMasking(BYTE u8Mask, BYTE u8Opacity)

Params

u8Mask	Show Masked Region 0 No 1 Yes
u8Opacity	Mask Opacity value Integer value 0, 20, 40, 60, 80 or 100

GetMarkers

Returns the markers setting

API
MONITOR_CODE GetMarkers(BYTE *pu8Val)

Params

*pu8Val	Pointer to return markers setting value

Return

pu8Val

```
typedef enum markers
{
    MARKERS_NONE = 0x00,
    MARKERS_1_85X1 = 0x01,
    MARKERS_2_39X1 = 0x02,
    MARKERS_2_35X1 = 0x03,
    MARKERS_1X1 = 0x04,
    MARKERS_16X9_EXTRACTION = 0x05,
    MARKERS_16X9_ACTION_SAFE = 0x06,
    MARKERS_16X9_TILE_SAFE = 0x07,
    MARKERS_4X3_EXTRACTION = 0x08,
    MARKERS_4X3_ACTION_SAFE = 0x09,
    MARKERS_4X3_TILE_SAFE = 0x0A,
    MARKERS_CENTER_CROSSHAIR = 0x0B,
    MARKERS_THIRDS = 0x0C,
    MARKERS_2_2X1 = 0x0D
}
MARKERS;
```

SetMarkers

Sets the markers setting

API
MONITOR_CODE SetMarkers(BYTE u8Val)

Params
u8Val

```
typedef enum markers
{
    MARKERS_NONE = 0x00,
    MARKERS_1_85X1 = 0x01,
    MARKERS_2_39X1 = 0x02,
    MARKERS_2_35X1 = 0x03,
    MARKERS_1X1 = 0x04,
    MARKERS_16X9_EXTRACTION = 0x05,
    MARKERS_16X9_ACTION_SAFE = 0x06,
    MARKERS_16X9_TILE_SAFE = 0x07,
    MARKERS_4X3_EXTRACTION = 0x08,
    MARKERS_4X3_ACTION_SAFE = 0x09,
    MARKERS_4X3_TILE_SAFE = 0x0A,
    MARKERS_CENTER_CROSSHAIR = 0x0B,
    MARKERS_THIRDS = 0x0C,
    MARKERS_2_2X1 = 0x0D
}
MARKERS;
```

GetMarkersColor

Returns the markers color setting

API
MONITOR_CODE GetMarkersColor(BYTE *pu8Val)

Params

*pu8Val	Pointer to return markers color setting value

Return
pu8Val

```
typedef enum markers_color
{
    MARKERS_COLOR_WHITE = 0,
    MARKERS_COLOR_RED = 1,
    MARKERS_COLOR_GREEN = 2,
    MARKERS_COLOR_BLUE = 3
}
MARKERS_COLOR;
```

SetMarkersColor

Sets the markers color setting

API
MONITOR_CODE SetMarkersColor(BYTE u8Val)

Params

u8Val	`typedef enum markers_color { MARKERS_COLOR_WHITE = 0, MARKERS_COLOR_RED = 1, MARKERS_COLOR_GREEN = 2, MARKERS_COLOR_BLUE = 3 } MARKERS_COLOR;`

```
typedef enum markers_color
{
    MARKERS_COLOR_WHITE = 0,
    MARKERS_COLOR_RED = 1,
    MARKERS_COLOR_GREEN = 2,
    MARKERS_COLOR_BLUE = 3
}
MARKERS_COLOR;
```

GetVideoDataRange

Returns the video data range

API
MONITOR_CODE GetVideoDataRange(BYTE *pu8Val)

Params

*pu8Val	Pointer to return video data range value

Return

pu8Val

```
typedef enum video_data_range
{
      VIDEO_DATA_RANGE_AUTO = 0,
      VIDEO_DATA_RANGE_FULL = 1,
      VIDEO_DATA_RANGE_LIMITED = 2
}
VIDEO_DATA_RANGE;
```

SetVideoDataRange

Sets the video data range

API
MONITOR_CODE SetVideoDataRange(BYTE u8Val)

Params

u8Val

```
typedef enum video_data_range
{
      VIDEO_DATA_RANGE_AUTO = 0,
      VIDEO_DATA_RANGE_FULL = 1,
      VIDEO_DATA_RANGE_LIMITED = 2
}
VIDEO_DATA_RANGE;
```

GetOverscanFrame

Returns if overscan frame by 5% is enabled

API
MONITOR_CODE GetOverscanFrame(BYTE *pu8Val)

Params

*pu8Val	Pointer to return if overscan frame by 5% is enabled

Return

pu8Val	Overscan frame by 5%	
	0	Off
	1	On

SetOverscanFrame

Enable/Disable overscan frame by 5%

API
MONITOR_CODE SetOverscanFrame(BYTE u8Val)

Params

u8Val	Overscan frame by 5%	
	0	Off
	1	On

GetBlueChannelOnly

Returns if blue channel only feature is enabled

API
MONITOR_CODE GetBlueChannelOnly(BYTE *pu8Val)

Params

*pu8Val	Pointer to return if blue channel only feature is enabled

Return

pu8Val	Blue Channel Only feature	
	0	Off
	1	On

SetBlueChannelOnly

Enable/Disable blue channel only feature

API
MONITOR_CODE SetBlueChannelOnly(BYTE u8Val)

Params

u8Val	Blue Channel Only feature	
	0	Off
	1	On

Color Management

GetSaturation

Returns the color saturation level

API

MONITOR_CODE GetSaturation(BYTE *pu8Val)

Params

*pu8Val	Pointer to return color saturation level

Return

pu8Val	Color Saturation level Integer value 0 to 100 Default 50 Values in increments of 1

SetSaturation

Sets the color saturation level

API

MONITOR_CODE SetSaturation(BYTE u8Val)

Params

u8Val	Color Saturation level Integer value 0 to 100 Default 50 Values in increments of 1

GetHue

Returns the hue level

API

MONITOR_CODE GetHue(BYTE *pu8Val)

Params

*pu8Val	Pointer to return hue level

Return

pu8Val	Color Saturation level Integer value 0 to 100 Default 50 Values in increments of 1

SetHue

Sets the hue level

API
MONITOR_CODE SetHue(UBYTE u8Val)

Params

u8Val	Hue level Integer value 0 to 100 Default 50 Values in increments of 1

GetColorTempCaps

Returns the supported color temperatures of the monitor

API
MONITOR_CODE GetColorTempCaps(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return color temperature capabilities

Return

pu32Val — Bitwise OR representation of color temperature capabilities

```
typedef enum color_temp
{
    COLOR_TEMP_5000K  = 0x00000001,
    COLOR_TEMP_5700K  = 0x00000002,
    COLOR_TEMP_6500K  = 0x00000004,
    COLOR_TEMP_7500K  = 0x00000008,
    COLOR_TEMP_9300K  = 0x00000010,
    COLOR_TEMP_10000K = 0x00000020
}
COLOR_TEMP;
```

For example:
0x00000013 would indicate 5000K, 5700K and 9300K supported

GetColorTemp

Returns the current color temperature

API
MONITOR_CODE GetColorTemp(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return color temperature

Return

pu32Val

```
typedef enum color_temp
{
    COLOR_TEMP_5000K  = 0x00000001,
    COLOR_TEMP_5700K  = 0x00000002,
    COLOR_TEMP_6500K  = 0x00000004,
    COLOR_TEMP_7500K  = 0x00000008,
    COLOR_TEMP_9300K  = 0x00000010,
    COLOR_TEMP_10000K = 0x00000020
}
COLOR_TEMP;
```

SetColorTemp

Sets the color temperature

API
MONITOR_CODE SetColorTemp(UWORD32 u32Val)

Params
u32Val

```
typedef enum color_temp
{
    COLOR_TEMP_5000K  = 0x00000001,
    COLOR_TEMP_5700K  = 0x00000002,
    COLOR_TEMP_6500K  = 0x00000004,
    COLOR_TEMP_7500K  = 0x00000008,
    COLOR_TEMP_9300K  = 0x00000010,
    COLOR_TEMP_10000K = 0x00000020
}
COLOR_TEMP;
```

GetColorSpaceCaps

Returns the supported color spaces of the monitor

API
MONITOR_CODE GetColorSpaceCaps(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return color space capabilities

Return

pu32Val	Bitwise OR representation of supported color spaces

```
typedef enum color_space
{
      /* Pre-UP2720Q */
      COLOR_SPACE_ADOBE_RGB       = 0x00000001,
      COLOR_SPACE_SRGB            = 0x00000002,
      COLOR_SPACE_REC_709         = 0x00000004,
      COLOR_SPACE_DCI_P3          = 0x00000008,
      COLOR_SPACE_CAL_1           = 0x00000010,
      COLOR_SPACE_CAL_2           = 0x00000020,
      COLOR_SPACE_REC_2020        = 0x00000040,

      /* UP2720Q */
      COLOR_SPACE2_DCI_P3         = 0x10000001,
      COLOR_SPACE2_BT_709         = 0x10000002,
      COLOR_SPACE2_BT_2020        = 0x10000004,
      COLOR_SPACE2_SRGB           = 0x10000008,
      COLOR_SPACE2_ADOBE_RGB_D65  = 0x10000010,
      COLOR_SPACE2_ADOBE_RGB_D50  = 0x10000020,
      COLOR_SPACE2_NATIVE         = 0x10000040,
      COLOR_SPACE2_CUSTOM_1       = 0x10000080,
      COLOR_SPACE2_CUSTOM_2       = 0x10000100,
      COLOR_SPACE2_CUSTOM_3       = 0x10000200,
      COLOR_SPACE2_CAL_1          = 0x10000400,
      COLOR_SPACE2_CAL_2          = 0x10000800
}
COLOR_SPACE;
```

GetColorSpaceState

Returns the current color space state

API
MONITOR_CODE GetColorSpaceState(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return color space state

Return
pu32Val

```
typedef enum color_space
{
        /* Pre-UP2720Q */
        COLOR_SPACE_ADOBE_RGB       = 0x00000001,
        COLOR_SPACE_SRGB            = 0x00000002,
        COLOR_SPACE_REC_709         = 0x00000004,
        COLOR_SPACE_DCI_P3          = 0x00000008,
        COLOR_SPACE_CAL_1           = 0x00000010,
        COLOR_SPACE_CAL_2           = 0x00000020,
        COLOR_SPACE_REC_2020        = 0x00000040,

        /* UP2720Q */
        COLOR_SPACE2_DCI_P3         = 0x10000001,
        COLOR_SPACE2_BT_709         = 0x10000002,
        COLOR_SPACE2_BT_2020        = 0x10000004,
        COLOR_SPACE2_SRGB           = 0x10000008,
        COLOR_SPACE2_ADOBE_RGB_D65  = 0x10000010,
        COLOR_SPACE2_ADOBE_RGB_D50  = 0x10000020,
        COLOR_SPACE2_NATIVE         = 0x10000040,
        COLOR_SPACE2_CUSTOM_1       = 0x10000080,
        COLOR_SPACE2_CUSTOM_2       = 0x10000100,
        COLOR_SPACE2_CUSTOM_3       = 0x10000200,
        COLOR_SPACE2_CAL_1          = 0x10000400,
        COLOR_SPACE2_CAL_2          = 0x10000800
}
COLOR_SPACE;
```

SetColorSpaceState

Sets the color space state

API
MONITOR_CODE SetColorSpaceState(UWORD32 u32Val)

Params
u32Val

```
typedef enum color_space
{
        /* Pre-UP2720Q */
        COLOR_SPACE_ADOBE_RGB       = 0x00000001,
        COLOR_SPACE_SRGB            = 0x00000002,
        COLOR_SPACE_REC_709          = 0x00000004,
        COLOR_SPACE_DCI_P3           = 0x00000008,
        COLOR_SPACE_CAL_1            = 0x00000010,
        COLOR_SPACE_CAL_2            = 0x00000020,
        COLOR_SPACE_REC_2020         = 0x00000040,

        /* UP2720Q */
        COLOR_SPACE2_DCI_P3          = 0x10000001,
        COLOR_SPACE2_BT_709          = 0x10000002,
        COLOR_SPACE2_BT_2020         = 0x10000004,
        COLOR_SPACE2_SRGB            = 0x10000008,
        COLOR_SPACE2_ADOBE_RGB_D65   = 0x10000010,
        COLOR_SPACE2_ADOBE_RGB_D50   = 0x10000020,
        COLOR_SPACE2_NATIVE          = 0x10000040,
        COLOR_SPACE2_CUSTOM_1        = 0x10000080,
        COLOR_SPACE2_CUSTOM_2        = 0x10000100,
        COLOR_SPACE2_CUSTOM_3        = 0x10000200,
        COLOR_SPACE2_CAL_1           = 0x10000400,
        COLOR_SPACE2_CAL_2           = 0x10000800
}
COLOR_SPACE;
```

GetInputColorFormat

Returns the input color format

API
MONITOR_CODE GetInputColorFormat(BYTE *pu8Val)

Params

*pu8Val	Pointer to return response time value

Return

pu8Val	Input Color Format
	0 RGB
	1 YPbPr

```
typedef enum input_color_format
{
      INPUT_COLOR_FORMAT_RGB   = 0,
      INPUT_COLOR_FORMAT_YPBPR = 1
}
INPUT_COLOR_FORMAT;
```

SetInputColorFormat

Sets the input color format

API
MONITOR_CODE SetInputColorFormat(BYTE u8Val)

Params

u8Val	Input Color Format
	0 RGB
	1 YPbPr

```
typedef enum input_color_format
{
      INPUT_COLOR_FORMAT_RGB   = 0,
      INPUT_COLOR_FORMAT_YPBPR = 1
}
INPUT_COLOR_FORMAT;
```

GetColorPresetCaps

Returns the available color presets

API
MONITOR_CODE GetColorPresetCaps(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return color space capabilities

Return

pu32Val — Bitwise OR representation of supported color presets

```
typedef enum color_preset
{
      COLOR_PRESET_STANDARD     = 0x00000001,
      COLOR_PRESET_MULTIMEDIA   = 0x00000002,
      COLOR_PRESET_MOVIE        = 0x00000004,
      COLOR_PRESET_GAME         = 0x00000008,
      COLOR_PRESET_PAPER        = 0x00000010,
      COLOR_PRESET_COLOR_TEMP   = 0x00000020,
      COLOR_PRESET_COLOR_SPACE  = 0x00000040,
      COLOR_PRESET_CUSTOM_COLOR = 0x00000080,
      COLOR_PRESET_DICOM        = 0x00000100,
      COLOR_PRESET_COMFORTVIEW  = 0x00000200,
      COLOR_PRESET_WARM         = 0x00000400,
      COLOR_PRESET_COOL         = 0x00000800,
      COLOR_PRESET_SRGB         = 0x00001000,
      COLOR_PRESET_GAME_FPS     = 0x00002000,
      COLOR_PRESET_GAME_RTS     = 0x00004000,
      COLOR_PRESET_GAME_RPG     = 0x00008000
}
COLOR_PRESET;
```

For example:
0x00000013 would indicate Standard, Multimedia and Paper presets available

GetColorPreset

Returns the current color preset

API
MONITOR_CODE GetColorPreset(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return color preset

Return

pu32Val

```
typedef enum color_preset
{
      COLOR_PRESET_STANDARD     = 0x00000001,
      COLOR_PRESET_MULTIMEDIA   = 0x00000002,
      COLOR_PRESET_MOVIE        = 0x00000004,
      COLOR_PRESET_GAME         = 0x00000008,
      COLOR_PRESET_PAPER        = 0x00000010,
      COLOR_PRESET_COLOR_TEMP   = 0x00000020,
      COLOR_PRESET_COLOR_SPACE  = 0x00000040,
      COLOR_PRESET_CUSTOM_COLOR = 0x00000080,
      COLOR_PRESET_DICOM        = 0x00000100,
      COLOR_PRESET_COMFORTVIEW  = 0x00000200,
      COLOR_PRESET_WARM         = 0x00000400,
      COLOR_PRESET_COOL         = 0x00000800,
      COLOR_PRESET_SRGB         = 0x00001000,
      COLOR_PRESET_GAME_FPS     = 0x00002000,
      COLOR_PRESET_GAME_RTS     = 0x00004000,
      COLOR_PRESET_GAME_RPG     = 0x00008000
}
COLOR_PRESET;
```

SetColorPreset

Sets the color preset

API
MONITOR_CODE SetColorPreset(UWORD32 u32Val)

Params
u32Val

```
typedef enum color_preset
{
      COLOR_PRESET_STANDARD     = 0x00000001,
      COLOR_PRESET_MULTIMEDIA   = 0x00000002,
      COLOR_PRESET_MOVIE        = 0x00000004,
      COLOR_PRESET_GAME         = 0x00000008,
      COLOR_PRESET_PAPER        = 0x00000010,
      COLOR_PRESET_COLOR_TEMP   = 0x00000020,
      COLOR_PRESET_COLOR_SPACE  = 0x00000040,
      COLOR_PRESET_CUSTOM_COLOR = 0x00000080,
      COLOR_PRESET_DICOM        = 0x00000100,
      COLOR_PRESET_COMFORTVIEW  = 0x00000200,
      COLOR_PRESET_WARM         = 0x00000400,
      COLOR_PRESET_COOL         = 0x00000800,
      COLOR_PRESET_SRGB         = 0x00001000,
      COLOR_PRESET_GAME_FPS     = 0x00002000,
      COLOR_PRESET_GAME_RTS     = 0x00004000,
      COLOR_PRESET_GAME_RPG     = 0x00008000
}
COLOR_PRESET;
```

GetCustomColor

Returns the current custom color. Ensure that the monitor is in the correct Color Preset or Color Space State first.

API
MONITOR_CODE GetCustomColor(BYTE u8Setting, BYTE *pu8ValR, BYTE *pu8ValG, BYTE *pu8ValB, BYTE *pu8ValC, BYTE *pu8ValM, BYTE *pu8ValY)

Params

u8Setting

```
typedef enum custom_color
{
    CUSTOM_COLOR_GAIN       = 0,
    CUSTOM_COLOR_OFFSET     = 1,
    CUSTOM_COLOR_HUE        = 2,
    CUSTOM_COLOR_SATURATION = 3,
    CUSTOM_COLOR_LIGHTNESS  = 4,
}
CUSTOM_COLOR;
```

*pu8ValR	Pointer to return R value
*pu8ValG	Pointer to return G value
*pu8ValB	Pointer to return B value
*pu8ValC	Pointer to return C value
*pu8ValM	Pointer to return M value
*pu8ValY	Pointer to return Y value

Return

pu8ValR	R value, 0 to 100
pu8ValG	G value, 0 to 100
pu8ValB	B value, 0 to 100
pu8ValC	C value, 0 to 100 (Only valid for custom color types Hue and Saturation)
pu8ValM	M value, 0 to 100 (Only valid for custom color types Hue and Saturation)
pu8ValY	Y value, 0 to 100 (Only valid for custom color types Hue and Saturation)

SetCustomColor

Sets the custom color. Ensure that the monitor is in the correct Color Preset or Color Space State first.

API
MONITOR_CODE SetCustomColor(BYTE u8Setting, BYTE u8ValR, BYTE u8ValG, BYTE u8ValB, BYTE u8ValC, BYTE u8ValM, BYTE u8ValY)

Params

u8Setting

```
typedef enum custom_color
{
    CUSTOM_COLOR_GAIN       = 0,
    CUSTOM_COLOR_OFFSET     = 1,
    CUSTOM_COLOR_HUE        = 2,
    CUSTOM_COLOR_SATURATION = 3,
    CUSTOM_COLOR_LIGHTNESS  = 4,
}
CUSTOM_COLOR;
```

u8ValR	R value, 0 to 100
u8ValG	G value, 0 to 100
u8ValB	B value, 0 to 100
u8ValC	C value, 0 to 100 (Only valid for custom color types Hue and Saturation)
u8ValM	M value, 0 to 100 (Only valid for custom color types Hue and Saturation)
u8ValY	Y value, 0 to 100 (Only valid for custom color types Hue and Saturation)

GetGammaMode

Returns the gamma mode

API
MONITOR_CODE GetGammaMode(BYTE *pu8Val)

Params

*pu8Val	Pointer to return gamma mode

Return

pu8Val	Gamma Mode	
	0	PC
	1	MAC

```
typedef enum gamma_mode
{
      GAMMA_MODE_PC  = 0,
      GAMMA_MODE_MAC = 1
}
GAMMA_MODE;
```

SetGammaMode

Sets the gamma mode

API
MONITOR_CODE SetGammaMode(BYTE u8Val)

Params

u8Val	Gamma Mode	
	0	PC
	1	MAC

```
typedef enum gamma_mode
{
      GAMMA_MODE_PC  = 0,
      GAMMA_MODE_MAC = 1
}
GAMMA_MODE;
```

GetUniformityCompensation

Returns the uniformity compensation setting

API
MONITOR_CODE GetUniformityCompensation(BYTE *pu8Val)

Params

*pu8Val	Pointer to return uniformity compensation setting

Return

pu8Val

```
typedef enum uniformity_compensation
{
      UNIFORMITY_COMPENSATION_OFF             = OFF,
      UNIFORMITY_COMPENSATION_ON              = 2,
      UNIFORMITY_COMPENSATION_CALIBRATED_HIGH = 2
}
UNIFORMITY_COMPENSATION;
```

SetUniformityCompensation

Sets the uniformity compensation

API
MONITOR_CODE SetUniformityCompensation(BYTE u8Val)

Params

u8Val

```
typedef enum uniformity_compensation
{
      UNIFORMITY_COMPENSATION_OFF             = OFF,
      UNIFORMITY_COMPENSATION_ON              = 2,
      UNIFORMITY_COMPENSATION_CALIBRATED_HIGH = 2
}
UNIFORMITY_COMPENSATION;
```

GetColorSpaceInfo

Returns the color space info for the color space mode as specified in the structure

API
MONITOR_CODE GetColorSpaceInfo(ColorSpaceInfoStructType *pData)

Params

* pData	Pointer to return color space info data

Return

pData — Set pData->ColorSpaceMode to retrieve info of the particular color space mode

```
typedef struct ColorSpaceInfoStruct {
    UWORD32 ColorSpaceMode; //enum COLOR_SPACE_MODE;
    FLOAT Coordinate_R[2]; //(x,y)
    FLOAT Coordinate_G[2]; //(x,y)
    FLOAT Coordinate_B[2]; //(x,y)
    FLOAT Coordinate_W[2]; //(x,y)
    BYTE GammaValue; //10-1Ah: 1.6-2.6, 0x20:bt1886,
    0x21:sRGB, 0x22:EPD, 0x24:EBU
    UWORD16 Luminance;
    BYTE stTargetCalibrationDate[5]; //mmhhDDMMYY
    BYTE stTargetValidationDate[5]; //mmhhDDMMYY
    BYTE stActualCalibrationDate[5]; //mmhhDDMMYY
    BYTE stActualValidationDate[5]; //mmhhDDMMYY
    BYTE CalibrationNow; //unused
    BYTE UniformityStatus; //0: OFF, 1: ON
    BYTE ColorBlocksState; //Bit[0]: PreGamma, Bit[1]:
    Matrix, Bit[2]: PostGamma, Bit[3]: 3DLut, Bit[4]:
    CalMAN Ready calibrated
    UWORD16 UsageHours; //READ-ONLY - Number of hours
    this color space mode is used after last
    calibration
    UWORD16 reserved;
}
ColorSpaceInfoStructType;
```

SetColorSpaceInfo

Set the color space info for the specified color space mode

API
MONITOR_CODE SetColorSpaceInfo(ColorSpaceInfoStructType *pData

Params

* pData	Pointer to return color space info data

Return

pData

```
typedef struct ColorSpaceInfoStruct {
    UWORD32 ColorSpaceMode; //enum COLOR_SPACE_MODE;
    FLOAT Coordinate_R[2]; //(x,y)
    FLOAT Coordinate_G[2]; //(x,y)
    FLOAT Coordinate_B[2]; //(x,y)
    FLOAT Coordinate_W[2]; //(x,y)
    BYTE GammaValue; //10-1Ah: 1.6-2.6, 0x20:bt1886,
    0x21:sRGB, 0x22:EPD, 0x24:EBU
    UWORD16 Luminance;
    BYTE stTargetCalibrationDate[5]; //mmhhDDMMYY
    BYTE stTargetValidationDate[5]; //mmhhDDMMYY
    BYTE stActualCalibrationDate[5]; //mmhhDDMMYY
    BYTE stActualValidationDate[5]; //mmhhDDMMYY
    BYTE CalibrationNow; //unused
    BYTE UniformityStatus; //0: OFF, 1: ON
    BYTE ColorBlocksState; //Bit[0]: PreGamma, Bit[1]:
    Matrix, Bit[2]: PostGamma, Bit[3]: 3DLut, Bit[4]:
    CalMAN Ready calibrated
    UWORD16 UsageHours; //READ-ONLY - Number of hours
    this color space mode is used after last
    calibration
    UWORD16 reserved;
}
ColorSpaceInfoStructType;
```

GetColorGamut

Returnd the current color gamut

API

MONITOR_CODE GetColorGamut(WORD32 *pu32Val)

Params

pu32Val	Pointer to return current color gamut

Return

*pu32Val

```
typedef enum color_gamut
{
      COLOR_GAMUT_DCI_P3  = 0x10000001,
      COLOR_GAMUT_BT_709  = 0x10000002,
      COLOR_GAMUT_BT_2020 = 0x10000004,
      COLOR_GAMUT_SRGB    = 0x10000008,
      COLOR_GAMUT_ADOBE   = 0x10000010,
      COLOR_GAMUT_NATIVE  = 0x10000040
}
COLOR_GAMUT;
```

SetColorGamut

Set current color gamut

API

MONITOR_CODE SetColorGamut(WORD32 u32Val)

Params

u32Val

```
typedef enum color_gamut
{
      COLOR_GAMUT_DCI_P3  = 0x10000001,
      COLOR_GAMUT_BT_709  = 0x10000002,
      COLOR_GAMUT_BT_2020 = 0x10000004,
      COLOR_GAMUT_SRGB    = 0x10000008,
      COLOR_GAMUT_ADOBE   = 0x10000010,
      COLOR_GAMUT_NATIVE  = 0x10000040
}
COLOR_GAMUT;
```

GetWhitePoint

Return the current white point

API

MONITOR_CODE GetWhitePoint(BYTE *pu8Val)

Params

pu8Val	Pointer to return current white point

Return

* pu8Val

```
typedef enum white_point
{
      WHITE_POINT_D50 = 1,
      WHITE_POINT_D55 = 2,
      WHITE_POINT_D60 = 3,
      WHITE_POINT_D65 = 4,
      WHITE_POINT_DCI_P3 = 5,
      WHITE_POINT_NATIVE = 6
}
WHITE_POINT;
```

SetWhitePoint

Set current white point

API

MONITOR_CODE SetWhitePoint(BYTE u8Val)

Params

u8Val

```
typedef enum white_point
{
      WHITE_POINT_D50 = 1,
      WHITE_POINT_D55 = 2,
      WHITE_POINT_D60 = 3,
      WHITE_POINT_D65 = 4,
      WHITE_POINT_DCI_P3 = 5,
      WHITE_POINT_NATIVE = 6
}
WHITE_POINT;
```

GetGamma

Return the current gamma

API

MONITOR_CODE GetGamma(BYTE *pu8Val)

Params

pu8Val	Pointer to return current gamma

Return

* pu8Val

```
typedef enum gamma
{
      GAMMA_1_6 = 0x01,
      GAMMA_1_8 = 0x02,
      GAMMA_2_0 = 0x03,
      GAMMA_2_2 = 0x04,
      GAMMA_2_4 = 0x05,
      GAMMA_2_6 = 0x06,
      GAMMA_BT_1886 = 0x07,
      GAMMA_SRGB = 0x08,
      GAMMA_NATIVE = 0x09
}
GAMMA;
```

SetGamma

Set current gamma

API

MONITOR_CODE SetGamma(BYTE u8Val)

Params

u8Val

```
typedef enum gamma
{
      GAMMA_1_6 = 0x01,
      GAMMA_1_8 = 0x02,
      GAMMA_2_0 = 0x03,
      GAMMA_2_2 = 0x04,
      GAMMA_2_4 = 0x05,
      GAMMA_2_6 = 0x06,
      GAMMA_BT_1886 = 0x07,
      GAMMA_SRGB = 0x08,
      GAMMA_NATIVE = 0x09
}
GAMMA;
```

GetLuminance

Return the current luminance level

API

MONITOR_CODE GetLuminance(UWORD16 *pu16Val)

Params

pu16Val	Pointer to return current luminance level

Return

* pu16Val	Luminance value Integer value 45 (dark) to 250 (bright) Values in increments of 1

SetLuminance

Set current luminance level

API

MONITOR_CODE SetLuminance(UWORD16 u16Val)

Params

u16Val	Luminance value Integer value 45 (dark) to 250 (bright) Values in increments of 1

GetCustomColorSpaceInfo

Returns the custom color space info for the custom color space as specified in the structure

API
MONITOR_CODE GetCustomColorSpaceInfo(CustomColorSpaceInfoStructType *pData)

Params

* pData	Pointer to return custom color space info data

Return

pData	Set pData->CustomColorSpace to retrieve info of the particular custom color space

```
typedef struct CustomColorSpaceInfoStruct {
    UWORD32 CustomColorSpace; //enum COLOR_SPACE
    UWORD32 ColorGamut;       //enum COLOR_GAMUT
    BYTE WhitePoint;          //enum WHITE_POINT
    BYTE Gamma;               //enum GAMMA
    UWORD16 Luminance;        //UP2720Q: 45-250
    BYTE Gain[3];             //[0-2] = R G B 0-100
    BYTE Offset[3];           //[0-2] = R G B 0-100
    BYTE Hue;                 //0-100
    BYTE Saturation;          //0-100
    BYTE SixAxis[6][3];       //[0-5]=[R G B C M Y] ,
    [0-2] = H S L 0-100
}
CustomColorSpaceInfoStructType;
```

ResetColor

Reset the current color space

API
MONITOR_CODE ResetColor(void)

Params
-

Video Input Management

GetAutoSelect

Returns the input source auto select setting

API
MONITOR_CODE GetAutoSelect(BYTE *pu8Val)

Params

*pu8Val	Pointer to return auto select setting

Return

pu8Val

```
typedef enum auto_select
{
      AUTO_SELECT_OFF    = OFF,
      AUTO_SELECT_ON     = ON,
      AUTO_SELECT_PROMPT = 2
}
AUTO_SELECT;
```

SetAutoSelect

Sets the input source auto select setting

API
MONITOR_CODE SetAutoSelect(BYTE u8Val)

Params

u8Val

```
typedef enum auto_select
{
      AUTO_SELECT_OFF    = OFF,
      AUTO_SELECT_ON     = ON,
      AUTO_SELECT_PROMPT = 2
}
AUTO_SELECT;
```

GetVideoInputCaps

Returns the available video inputs

API
MONITOR_CODE GetVideoInputCaps(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return available video inputs

Return

pu32Val — Bitwise OR representation of available video inputs

```
typedef enum video_input
{
    VIDEO_INPUT_HDMI1 = 0x0000001,
    VIDEO_INPUT_HDMI2 = 0x0000002,
    VIDEO_INPUT_HDMI3 = 0x0000004,
    VIDEO_INPUT_DP1   = 0x0000008,
    VIDEO_INPUT_DP2   = 0x0000010,
    VIDEO_INPUT_DP3   = 0x0000020,
    VIDEO_INPUT_VGA1  = 0x0000040,
    VIDEO_INPUT_VGA2  = 0x0000080,
    VIDEO_INPUT_DVI1  = 0x0000100,
    VIDEO_INPUT_DVI2  = 0x0000200,
    VIDEO_INPUT_TB1   = 0x0000400,
    VIDEO_INPUT_TB2   = 0x0000800
}
VIDEO_INPUT;
```

For example:
0x00000149 would indicate HDMI1, DP1, VGA1 and DVI1 available

GetVideoInput

Returns the current video input source

API
MONITOR_CODE GetVideoInput(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return video input source

Return

pu32Val

```
typedef enum video_input
{
    VIDEO_INPUT_HDMI1 = 0x0000001,
    VIDEO_INPUT_HDMI2 = 0x0000002,
    VIDEO_INPUT_HDMI3 = 0x0000004,
    VIDEO_INPUT_DP1   = 0x0000008,
    VIDEO_INPUT_DP2   = 0x0000010,
    VIDEO_INPUT_DP3   = 0x0000020,
    VIDEO_INPUT_VGA1  = 0x0000040,
    VIDEO_INPUT_VGA2  = 0x0000080,
    VIDEO_INPUT_DVI1  = 0x0000100,
    VIDEO_INPUT_DVI2  = 0x0000200,
    VIDEO_INPUT_TB1   = 0x0000400,
    VIDEO_INPUT_TB2   = 0x0000800
}
VIDEO_INPUT;
```

SetVideoInput

Sets the video input source

API
MONITOR_CODE SetVideoInput(UWORD32 u32Val)

Params

u32Val

```
typedef enum video_input
{
    VIDEO_INPUT_HDMI1 = 0x0000001,
    VIDEO_INPUT_HDMI2 = 0x0000002,
    VIDEO_INPUT_HDMI3 = 0x0000004,
    VIDEO_INPUT_DP1   = 0x0000008,
    VIDEO_INPUT_DP2   = 0x0000010,
    VIDEO_INPUT_DP3   = 0x0000020,
    VIDEO_INPUT_VGA1  = 0x0000040,
    VIDEO_INPUT_VGA2  = 0x0000080,
    VIDEO_INPUT_DVI1  = 0x0000100,
    VIDEO_INPUT_DVI2  = 0x0000200,
    VIDEO_INPUT_TB1   = 0x0000400,
    VIDEO_INPUT_TB2   = 0x0000800
}
VIDEO_INPUT;
```

GetVideoInputName

Returns the current video input name

API
MONITOR_CODE GetVideoInputName (UWORD32 u32VideoInput, BYTE *pu8Name)

Params

u32VideoInput — Video input source

```
typedef enum video_input
{
      VIDEO_INPUT_HDMI1 = 0x0000001,
      VIDEO_INPUT_HDMI2 = 0x0000002,
      VIDEO_INPUT_HDMI3 = 0x0000004,
      VIDEO_INPUT_DP1   = 0x0000008,
      VIDEO_INPUT_DP2   = 0x0000010,
      VIDEO_INPUT_DP3   = 0x0000020,
      VIDEO_INPUT_VGA1  = 0x0000040,
      VIDEO_INPUT_VGA2  = 0x0000080,
      VIDEO_INPUT_DVI1  = 0x0000100,
      VIDEO_INPUT_DVI2  = 0x0000200,
      VIDEO_INPUT_TB1   = 0x0000400,
      VIDEO_INPUT_TB2   = 0x0000800
}
VIDEO_INPUT;
```

*pu8Name — Pointer to return video input name

Return

pu8Name

```
typedef enum video_input_name
{
    VIDEO_INPUT_NAME_OFF      = 0,
    VIDEO_INPUT_NAME_PC       = 1,
    VIDEO_INPUT_NAME_PC_1     = 2,
    VIDEO_INPUT_NAME_PC_2     = 3,
    VIDEO_INPUT_NAME_LAPTOP   = 4,
    VIDEO_INPUT_NAME_LAPTOP_1 = 5,
    VIDEO_INPUT_NAME_LAPTOP_2 = 6,
}
VIDEO_INPUT_NAME;
```

SetVideoInputName

Sets the video input name

API
MONITOR_CODE SetVideoInputName(UWORD32 u32VideoInput, BYTE u8Name)

Params

u32VideoInput

```
typedef enum video_input
{
      VIDEO_INPUT_HDMI1 = 0x0000001,
      VIDEO_INPUT_HDMI2 = 0x0000002,
      VIDEO_INPUT_HDMI3 = 0x0000004,
      VIDEO_INPUT_DP1   = 0x0000008,
      VIDEO_INPUT_DP2   = 0x0000010,
      VIDEO_INPUT_DP3   = 0x0000020,
      VIDEO_INPUT_VGA1  = 0x0000040,
      VIDEO_INPUT_VGA2  = 0x0000080,
      VIDEO_INPUT_DVI1  = 0x0000100,
      VIDEO_INPUT_DVI2  = 0x0000200,
      VIDEO_INPUT_TB1   = 0x0000400,
      VIDEO_INPUT_TB2   = 0x0000800
}
VIDEO_INPUT;
```

u8Name

```
typedef enum video_input_name
{
    VIDEO_INPUT_NAME_OFF      = 0,
    VIDEO_INPUT_NAME_PC       = 1,
    VIDEO_INPUT_NAME_PC_1     = 2,
    VIDEO_INPUT_NAME_PC_2     = 3,
    VIDEO_INPUT_NAME_LAPTOP   = 4,
    VIDEO_INPUT_NAME_LAPTOP_1 = 5,
    VIDEO_INPUT_NAME_LAPTOP_2 = 6,
}
VIDEO_INPUT_NAME;
```

GetAutoSelectTbt

Returns the current setting for auto select of Thunderbolt inputs

API

MONITOR_CODE GetAutoSelectTbt(BYTE *pu8Val)

Params

*pu8Val	Pointer to return auto select setting

Return

pu8Name

```
typedef enum auto_select
{
    AUTO_SELECT_OFF    = OFF,
    AUTO_SELECT_ON     = ON,
    AUTO_SELECT_PROMPT = 2
}
AUTO_SELECT;
```

SetAutoSelectTbt

Sets the auto select of Thunderbolt inputs

API
MONITOR_CODE SetAutoSelectTbt(BYTE u8Val)

Params

u8Val	see below

u8Val

```
typedef enum auto_select
{
    AUTO_SELECT_OFF    = OFF,
    AUTO_SELECT_ON     = ON,
    AUTO_SELECT_PROMPT = 2
}
AUTO_SELECT;
```

PIP/PBP Management

GetPxPMode

Returns the current PIP/PBP mode

API
MONITOR_CODE GetPxPMode(BYTE *pu8Val)

Params

*pu8Val	Pointer to return PIP/PBP mode

Return

pu8Val

```
typedef enum pxp_mode
{
    PXP_OFF              = 0,
    PXP_PIP_SMALL        = 1,
    PXP_PIP_LARGE        = 2,
    PXP_PBP_ASPECT_RATIO = 3,
    PXP_PBP_FILL         = 4,
    PXP_PBP_AA           = 5,
    PXP_PBP_AB           = 6
}
PXP_MODE;
```

SetPxPMode

Sets the PIP/PBP mode

API
MONITOR_CODE SetPxPMode(BYTE u8Val)

Params

u8Val

```
typedef enum pxp_mode
{
    PXP_OFF              = 0,
    PXP_PIP_SMALL        = 1,
    PXP_PIP_LARGE        = 2,
    PXP_PBP_ASPECT_RATIO = 3,
    PXP_PBP_FILL         = 4,
    PXP_PBP_AA           = 5,
    PXP_PBP_AB           = 6
}
PXP_MODE;
```

GetPxPSubInput

Returns the current PxP sub video input source

API
MONITOR_CODE GetPxPSubInput(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return PxP sub video input source

Return
pu32Val

```
typedef enum video_input
{
    VIDEO_INPUT_HDMI1 = 0x0000001,
    VIDEO_INPUT_HDMI2 = 0x0000002,
    VIDEO_INPUT_HDMI3 = 0x0000004,
    VIDEO_INPUT_DP1   = 0x0000008,
    VIDEO_INPUT_DP2   = 0x0000010,
    VIDEO_INPUT_DP3   = 0x0000020,
    VIDEO_INPUT_VGA1  = 0x0000040,
    VIDEO_INPUT_VGA2  = 0x0000080,
    VIDEO_INPUT_DVI1  = 0x0000100,
    VIDEO_INPUT_DVI2  = 0x0000200,
    VIDEO_INPUT_TB1   = 0x0000400,
    VIDEO_INPUT_TB2   = 0x0000800
}
VIDEO_INPUT;
```

SetPxPSubInput

Sets the PxP sub video input source

API
MONITOR_CODE SetPxPSubInput(UWORD32 u32Val)

Params
u32Val

```
typedef enum video_input
{
    VIDEO_INPUT_HDMI1 = 0x0000001,
    VIDEO_INPUT_HDMI2 = 0x0000002,
    VIDEO_INPUT_HDMI3 = 0x0000004,
    VIDEO_INPUT_DP1   = 0x0000008,
    VIDEO_INPUT_DP2   = 0x0000010,
    VIDEO_INPUT_DP3   = 0x0000020,
    VIDEO_INPUT_VGA1  = 0x0000040,
    VIDEO_INPUT_VGA2  = 0x0000080,
    VIDEO_INPUT_DVI1  = 0x0000100,
    VIDEO_INPUT_DVI2  = 0x0000200,
    VIDEO_INPUT_TB1   = 0x0000400,
    VIDEO_INPUT_TB2   = 0x0000800
}
VIDEO_INPUT;
```

GetPxPLocation

Returns the current PxP location

API
MONITOR_CODE GetPxPLocation(BYTE *pu8Val)

Params

*pu8Val	Pointer to return PxP sub video input source

Return

pu8Val

```
typedef enum pxp_pip_location
{
    PXP_PIP_LOCATION_TOP_RIGHT    = 0,
    PXP_PIP_LOCATION_TOP_LEFT     = 1,
    PXP_PIP_LOCATION_BOTTOM_RIGHT = 2,
    PXP_PIP_LOCATION_BOTTOM_LEFT  = 3
}
PXP_PIP_LOCATION;
```

SetPxPLocation

Sets the PxP location

API
MONITOR_CODE SetPxPLocation(BYTE *pu8Val)

Params

pu8Val

```
typedef enum pxp_pip_location
{
    PXP_PIP_LOCATION_TOP_RIGHT    = 0,
    PXP_PIP_LOCATION_TOP_LEFT     = 1,
    PXP_PIP_LOCATION_BOTTOM_RIGHT = 2,
    PXP_PIP_LOCATION_BOTTOM_LEFT  = 3
}
PXP_PIP_LOCATION;
```

GetPxPColorGamut

Returns the PxP color gamut

API
MONITOR_CODE GetPxPColorGamut(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return PxP color gamut

Return

pu32Val

```
typedef enum color_gamut
{
      COLOR_GAMUT_DCI_P3  = 0x10000001,
      COLOR_GAMUT_BT_709  = 0x10000002,
      COLOR_GAMUT_BT_2020 = 0x10000004,
      COLOR_GAMUT_SRGB    = 0x10000008,
      COLOR_GAMUT_ADOBE   = 0x10000010,
      COLOR_GAMUT_NATIVE  = 0x10000040
}
COLOR_GAMUT;
```

SetPxPColorGamut

Sets the PxP color gamut

API
MONITOR_CODE SetPxPColorGamut(UWORD32 u32Val)

Params

u32Val

```
typedef enum color_gamut
{
      COLOR_GAMUT_DCI_P3  = 0x10000001,
      COLOR_GAMUT_BT_709  = 0x10000002,
      COLOR_GAMUT_BT_2020 = 0x10000004,
      COLOR_GAMUT_SRGB    = 0x10000008,
      COLOR_GAMUT_ADOBE   = 0x10000010,
      COLOR_GAMUT_NATIVE  = 0x10000040
}
COLOR_GAMUT;
```

GetPxPColorGamma

Returns the current PxP color gamma

API
MONITOR_CODE GetPxPColorGamma(BYTE *pu8Val)

Params

*pu8Val	Pointer to return PxP color gamma

Return

pu8Val

```
typedef enum gamma
{
    GAMMA_1_6      = 0x01,
    GAMMA_1_8      = 0x02,
    GAMMA_2_0      = 0x03,
    GAMMA_2_2      = 0x04,
    GAMMA_2_4      = 0x05,
    GAMMA_2_6      = 0x06,
    GAMMA_BT_1886  = 0x07,
    GAMMA_SRGB     = 0x08,
    GAMMA_NATIVE   = 0x09
}
GAMMA;
```

SetPxPColorGamma

Sets the PxP color gamma

API
MONITOR_CODE SetPxPColorGamma(BYTE u8Val)

Params

u8Val

```
typedef enum gamma
{
    GAMMA_1_6      = 0x01,
    GAMMA_1_8      = 0x02,
    GAMMA_2_0      = 0x03,
    GAMMA_2_2      = 0x04,
    GAMMA_2_4      = 0x05,
    GAMMA_2_6      = 0x06,
    GAMMA_BT_1886  = 0x07,
    GAMMA_SRGB     = 0x08,
    GAMMA_NATIVE   = 0x09
}
GAMMA;
```

GetPxPWhitePoint

Returns the current PxP white point

API
MONITOR_CODE GetPxPWhitePoint(BYTE *pu8Val)

Params

*pu8Val	Pointer to return PxP white point

Return

pu8Val

```
typedef enum white_point
{
    WHITE_POINT_D50 = 1,
    WHITE_POINT_D55 = 2,
    WHITE_POINT_D60 = 3,
    WHITE_POINT_D65 = 4,
    WHITE_POINT_DCI_P3 = 5,
    WHITE_POINT_NATIVE = 6
}
WHITE_POINT;
```

SetPxPWhitePoint

Sets the PxP white point

API
MONITOR_CODE SetPxPWhitePoint(BYTE u8Val)

Params

u8Val

```
typedef enum white_point
{
    WHITE_POINT_D50 = 1,
    WHITE_POINT_D55 = 2,
    WHITE_POINT_D60 = 3,
    WHITE_POINT_D65 = 4,
    WHITE_POINT_DCI_P3 = 5,
    WHITE_POINT_NATIVE = 6
}
WHITE_POINT;
```

GetPxPSharpness

Returns the current PxP sharpness

API
MONITOR_CODE GetPxPSharpness(BYTE *pu8Val)

Params

*pu8Val	Pointer to return PxP sharpness level

Return

pu8Val	PxP sharpness level

SetPxPSharpness

Sets the PxP sharpness

API
MONITOR_CODE SetPxPSharpness(BYTE u8Val)

Params

u8Val	PxP sharpness level

GetPxPAudio

Returns the current PxP audio

API
MONITOR_CODE GetPxPAudio(BYTE *pu8Val)

Params

*pu8Val	Pointer to return PxP audio

Return

pu8Val

```
typedef enum audio_source
{
    AUDIO_SOURCE_MAIN = 0,
    AUDIO_SOURCE_SUB = 1
}
AUDIO_SOURCE;
```

SetPxPAudio

Sets the PxP audio

API
MONITOR_CODE SetPxPAudio(BYTE u8Val)

Params

u8Val

```
typedef enum audio_source
{
    AUDIO_SOURCE_MAIN = 0,
    AUDIO_SOURCE_SUB = 1
}
AUDIO_SOURCE;
```

GetPxPVideoRange

Returns the current PxP video range

API
MONITOR_CODE GetPxPVideoRange(BYTE *pu8Val)

Params

*pu8Val	Pointer to return PxP video range

Return

pu8Val

```
typedef enum video_data_range
{
    VIDEO_DATA_RANGE_AUTO = 0,
    VIDEO_DATA_RANGE_FULL = 1,
    VIDEO_DATA_RANGE_LIMITED = 2
}
VIDEO_DATA_RANGE;
```

SetPxPVideoRange

Sets the PxP video range

API
MONITOR_CODE SetPxPVideoRange(BYTE u8Val)

Params

u8Val

```
typedef enum video_data_range
{
    VIDEO_DATA_RANGE_AUTO = 0,
    VIDEO_DATA_RANGE_FULL = 1,
    VIDEO_DATA_RANGE_LIMITED = 2
}
VIDEO_DATA_RANGE;
```

PxPInputToggle

Returns the current PxP input toggle

API
MONITOR_CODE PxPInputToggle(void)

Params
-

PxPVideoSwap

Sets the PxP video swap

API
MONITOR_CODE PxPVideoSwap(void)

Params
-

OSD Management

GetOSDTransparency

Get the OSD Transparency

API
MONITOR_CODE GetOSDTransparency(BYTE *pu8Val)

Params

*pu8Val	Pointer to return OSD Transparency value

Return

pu8Val	OSD Transparency Integer value 0 (opaque) to 100 (transparent) Default 20 Values in increments of 20

SetOSDTransparency

Set the OSD Transparency

API
MONITOR_CODE SetOSDTransparency(BYTE u8Val)

Params

u8Val	OSD Transparency Integer value 0 (opaque) to 100 (transparent) Default 20 Values in increments of 20

GetOSDLanguage

Get the OSD Language

API
MONITOR_CODE GetOSDLanguage(BYTE *pu8Val)

Params

*pu8Val	Pointer to return OSD Language value

Return

pu8Val

```
typedef enum osd_language
{
      OSD_LANGUAGE_ENGLISH                = 0,
      OSD_LANGUAGE_ESPANOL                = 1,
      OSD_LANGUAGE_FRANCAIS               = 2,
      OSD_LANGUAGE_DEUTSCH                = 3,
      OSD_LANGUAGE_PORTUGUES_BRASIL       = 4,
      OSD_LANGUAGE_PYCCKNN                = 5,
      OSD_LANGUAGE_CHINESE_SIMPLIFIED     = 6,
      OSD_LANGUAGE_JAPANESE               = 7
}
OSD_LANGUAGE;
```

SetOSDLanguage

Set the OSD Language

API

MONITOR_CODE SetOSDLanguage(BYTE u8Val)

Params

u8Val

```
typedef enum osd_language
{
      OSD_LANGUAGE_ENGLISH                  = 0,
      OSD_LANGUAGE_ESPANOL                  = 1,
      OSD_LANGUAGE_FRANCAIS                 = 2,
      OSD_LANGUAGE_DEUTSCH                  = 3,
      OSD_LANGUAGE_PORTUGUES_BRASIL         = 4,
      OSD_LANGUAGE_PYCCKNN                  = 5,
      OSD_LANGUAGE_CHINESE_SIMPLIFIED       = 6,
      OSD_LANGUAGE_JAPANESE                 = 7
}
OSD_LANGUAGE;
```

GetOSDRotation

Get the OSD Rotation

API

MONITOR_CODE GetOSDRotation(BYTE *pu8Val)

Params

*pu8Val	Pointer to return OSD Rotation value

Return

pu8Val

```
typedef enum osd_rotation
{
    OSD_ROTATION_0         = 0,
    OSD_ROTATION_90        = 1,
    OSD_ROTATION_270       = 2,
    OSD_ROTATION_180       = 3,
    OSD_ROTATION_AUTO_ON   = 4,
    OSD_ROTATION_AUTO_OFF  = 5
}
OSD_ROTATION;
```

SetOSDRotation

Set the OSD Rotations

API

MONITOR_CODE SetOSDRotation(BYTE u8Val)

Params

u8Val

```
typedef enum osd_rotation
{
    OSD_ROTATION_0         = 0,
    OSD_ROTATION_90        = 1,
    OSD_ROTATION_270       = 2,
    OSD_ROTATION_180       = 3,
    OSD_ROTATION_AUTO_ON   = 4,
    OSD_ROTATION_AUTO_OFF  = 5
}
OSD_ROTATION;
```

GetOSDTimer

Get the OSD Timer

API
MONITOR_CODE GetOSDTimer(BYTE *pu8Val)

Params

*pu8Val	Pointer to return OSD Timer value

Return

pu8Val	OSD Timer Integer value 5 to 60 seconds Default 20 seconds Values in increments of 1

SetOSDTimer

Set the OSD Timer

API
MONITOR_CODE SetOSDTimer(BYTE u8Val)

Params

u8Val	OSD Timer Integer value 5 to 60 seconds Default 20 seconds Values in increments of 1

GetOSDButtonLock

Get the OSD button lock.

API
MONITOR_CODE GetOSDButtonLock(BYTE *pu8Val)

Params

*pu8Val	Pointer to return OSD button lock

Return

pu8Val

```
typedef enum osd_button
{
    OSD_BUTTON_UNLOCK                  = 0,
    OSD_BUTTON_LOCK                    = 1,
    OSD_BUTTON_LOCK_OSD                = 1, // Menu
    Buttons
    OSD_BUTTON_LOCK_POWER              = 2, //Power
    Button
    OSD_BUTTON_LOCK_OSD_POWER          = 3, //Menu
    + Power Button
    OSD_BUTTON_LOCK_COLOR_CUSTOM_SETTINGS = 4 //Color
    Custom Settings
}
OSD_BUTTON;
```

SetOSDButtonLock

Set the OSD button lock

API
MONITOR_CODE SetOSDButtonLock(BYTE u8Val)

Params
u8Val

```
typedef enum osd_button
{
    OSD_BUTTON_UNLOCK                     = 0,
    OSD_BUTTON_LOCK                       = 1,
    OSD_BUTTON_LOCK_OSD                   = 1, // Menu
    Buttons
    OSD_BUTTON_LOCK_POWER                 = 2, //Power
    Button
    OSD_BUTTON_LOCK_OSD_POWER             = 3, //Menu +
    Power Button
    OSD_BUTTON_LOCK_COLOR_CUSTOM_SETTINGS = 4 //Color
    Custom Settings
}
OSD_BUTTON;
```

GetButtonSound

Returns if the button sound is on or off

API
MONITOR_CODE GetButtonSound(BYTE *pu8Val)

Params

*pu8Val	Pointer to return Button Sound value

Return

pu8Val — Button Sound

0	Off
1	On

SetButtonSound

Set the button sound on or off

API
MONITOR_CODE SetButtonSound(BYTE u8Val)

Params

u8Val — Button Sound

0	Off
1	On

System Management

GetVersionFirmware

Returns the firmware version of the monitor

API

MONITOR_CODE GetVersionFirmware(BYTE *pbyFirmwareVersion)

Params

*pbyFirmwareVersion	Pointer to firmware version for return

Return

pbyFirmwareVersion	Version string (max 10 chars)

GetVersionSDK

Returns the SDK version

API

MONITOR_CODE GetVersionSDK(UWORD16 *pu16Val)

Params

*pu16Val	Pointer to firmware version for return

Return

pu16Val	Version value where MSB = major version and LSB = minor version. Eg) 0x0100 will mean Version 1.0

GetMST

Returns if the MST is on or off

API

MONITOR_CODE GetMST(BYTE *pu8Val)

Params

*pu8Val	Pointer to return MST value

Return

pu8Val	MST value 0 Off 1 On

SetMST

Turns on / off the MST

API

MONITOR_CODE SetMST(BYTE u8Val)

Params

u8Val	MST value to set 0 Off 1 On

GetLCDConditioning

Returns if the LCD Conditioning is enabled or disabled

API

MONITOR_CODE GetLCDConditioning(BYTE *pu8Val)

Params

*pu8Val	Pointer to return LCD Conditioning value

Return

pu8Val	LCD Conditioning 0 Disabled 1 Enabled

SetLCDConditioning

Enable / Disable the LCD Conditioning

API

MONITOR_CODE SetLCDConditioning(BYTE u8Val)

Params

u8Val	LCD Conditioning value to set 0 Disable 1 Enable

FactoryReset

Reset to factory settings

API

MONITOR_CODE FactoryReset(void)

Params

-

SetDebugLevel

Set the level of debug for the SDK

API
MONITOR_CODE SetDebugLevel(BYTE u8Val)

Params

u8Val	`typedef enum dblevel { DB_OFF = 0, DB_ERROR = 1, DB_WARN = 2, DB_DEBUG = 3, DB_TRACE = 4 } DBLEVEL;`

```
typedef enum dblevel
{
      DB_OFF   = 0,
      DB_ERROR = 1,
      DB_WARN  = 2,
      DB_DEBUG = 3,
      DB_TRACE = 4
}
DBLEVEL;
```

KeepAlive

Keeps the session alive. Otherwise, session will be automatically terminated 300 seconds after the last command to the monitor.

API
MONITOR_CODE KeepAlive(void)

Params
-

GetDateTime

Returns date time
MONITOR_CODE GetDateTime(struct tm *pData)

Params

*pData	Pointer to return monitor's date and time

Return

pData	Monitor's date and time

SetDateTime

Set date time

API
MONITOR_CODE SetDateTime(struct tm *pData)

Params

*pData	Pointer to date and time data structure to set the monitor

GetAutoSleep

Returns auto sleep

API

MONITOR_CODE GetAutoSleep (BYTE *pu8Val)

Params

*pu8Val	Pointer to return auto sleep value

Return

pu8Val

```
typedef enum auto_sleep
{
    AUTO_SLEEP_DISPLAY   = 1,
    AUTO_SLEEP_PANEL_OFF = 2
}
AUTO_SLEEP;
```

SetAutoSleep

Set auto sleep

API

MONITOR_CODE SetAutoSleep (BYTE u8Val)

Params

u8Val

```
typedef enum auto_sleep
{
    AUTO_SLEEP_DISPLAY   = 1,
    AUTO_SLEEP_PANEL_OFF = 2
}
AUTO_SLEEP;
```

GetWarmUpTime

Returns warm up time

API

MONITOR_CODE GetWarmUpTime(BYTE *pu8Val, BYTE *pu8Day, BYTE *pu8Hour, BYTE *pu8Min)

Params

*pu8Val	Pointer to return warm up value
*pu8Day	Pointer to return day
*pu8Hour	Pointer to return hour
*pu8Min	Pointer to return minute

Return

pu8Val — Warm up value

pu8Day

```
typedef enum day_selection
{
    DAY_SELECTION_MON_FRI = 1,
```

	`    DAY_SELECTION_SAT_SUN = 2,`
	`    DAY_SELECTION_DAILY   = 3`
	`}`
	`DAY_SELECTION;`

pu8Hour	Hour
pu8Min	Minute

SetWarmUpTime

Set warm up time

API

MONITOR_CODE SetWarmUpTime(BYTE u8Val, BYTE u8Day, BYTE u8Hour, BYTE u8Min)

Params

u8Val	Warm up value

u8Day	`typedef enum day_selection`
	`{`
	`    DAY_SELECTION_MON_FRI = 1,`
	`    DAY_SELECTION_SAT_SUN = 2,`
	`    DAY_SELECTION_DAILY   = 3`
	`}`
	`DAY_SELECTION;`

u8Hour	Hour
u8Min	Minute

GetSoftwareLock

Get the software lock. Software lock will lock the various buttons independent of the OSD lock.

API

MONITOR_CODE GetSoftwareLock(BYTE *pu8Val)

Params

*pu8Val	Pointer to return software lock

Return

pu8Val

```
typedef enum software_lock
{
    SOFTWARE_LOCK_UNLOCK = OSD_BUTTON_UNLOCK,
    //Unlock all Locks
    SOFTWARE_LOCK_MENU = OSD_BUTTON_LOCK_OSD,
    //Lock Menu Buttons
    SOFTWARE_LOCK_POWER = OSD_BUTTON_LOCK_POWER,
    //Lock Power Button
    SOFTWARE_LOCK_MENU_POWER =
    OSD_BUTTON_LOCK_OSD_POWER,                 //Lock
    Menu + Power Button
    SOFTWARE_LOCK_COLOR_SETTINGS =
    OSD_BUTTON_LOCK_COLOR_CUSTOM_SETTINGS, //Lock
    Color Custom Settings
    SOFTWARE_LOCK_EXCEPT_POWER =
    OSD_BUTTON_LOCK_EXCEPT_POWER             //Lock
    all except Power Button
}
SOFTWARE_LOCK;
```

SetSoftwareLock

Set the software lock. Software lock will lock the various buttons independent of the OSD lock.

API

MONITOR_CODE SetSoftwareLock(BYTE u8Val)

Params

u8Val

```
typedef enum software_lock
{
    SOFTWARE_LOCK_UNLOCK = OSD_BUTTON_UNLOCK,
    //Unlock all Locks
    SOFTWARE_LOCK_MENU = OSD_BUTTON_LOCK_OSD,
    //Lock Menu Buttons
    SOFTWARE_LOCK_POWER = OSD_BUTTON_LOCK_POWER,
    //Lock Power Button
    SOFTWARE_LOCK_MENU_POWER =
    OSD_BUTTON_LOCK_OSD_POWER,                 //Lock
    Menu + Power Button
    SOFTWARE_LOCK_COLOR_SETTINGS =
    OSD_BUTTON_LOCK_COLOR_CUSTOM_SETTINGS, //Lock Color
    Custom Settings
    SOFTWARE_LOCK_EXCEPT_POWER =
    OSD_BUTTON_LOCK_EXCEPT_POWER             //Lock all
    except Power Button
}
SOFTWARE_LOCK;
```

ResetMenu

Returns reset menu value

API

MONITOR_CODE ResetMenu(BYTE u8Val)

Params

u8Val

```
typedef enum reset_menu
{
    RESET_MENU_POWER             = 0x01, /*replace
ResetPower */
    RESET_MENU_COLOR             = 0x02,
    RESET_MENU_OSD               = 0x03, /*replace
ResetOSD */
    RESET_MENU_COLORSPACE        = 0x04,
    RESET_MENU_INPUTSOURCE       = 0x05,
    RESET_MENU_DISPLAY           = 0x06,
    RESET_MENU_PXP               = 0x07,
    RESET_MENU_PERSONALIZATION = 0x08,
    RESET_MENU_OTHERS            = 0xFF
}
RESET_MENU;
```

Calibration Validation – OSD

GetCalibrationTarget

Return calibration targets as set in the monitor

API
MONITOR_CODE GetCalibrationTarget(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return calibration target value

Return

pu32Val

```
typedef enum color_space
{
    /* Pre-UP2720Q */
    COLOR_SPACE_ADOBE_RGB        = 0x00000001,
    COLOR_SPACE_SRGB             = 0x00000002,
    COLOR_SPACE_REC_709          = 0x00000004,
    COLOR_SPACE_DCI_P3           = 0x00000008,
    COLOR_SPACE_CAL_1            = 0x00000010,
    COLOR_SPACE_CAL_2            = 0x00000020,
    COLOR_SPACE_REC_2020         = 0x00000040,

    /* UP2720Q */
    COLOR_SPACE2_DCI_P3          = 0x10000001,
    COLOR_SPACE2_BT_709          = 0x10000002,
    COLOR_SPACE2_BT_2020         = 0x10000004,
    COLOR_SPACE2_SRGB            = 0x10000008,
    COLOR_SPACE2_ADOBE_RGB_D65   = 0x10000010,
    COLOR_SPACE2_ADOBE_RGB_D50   = 0x10000020,
    COLOR_SPACE2_NATIVE          = 0x10000040,
    COLOR_SPACE2_CUSTOM_1        = 0x10000080,
    COLOR_SPACE2_CUSTOM_2        = 0x10000100,
    COLOR_SPACE2_CUSTOM_3        = 0x10000200,
    COLOR_SPACE2_CAL_1           = 0x10000400,
    COLOR_SPACE2_CAL_2           = 0x10000800
}
COLOR_SPACE;
```

SetCalibrationTarget

Set calibration targets in the monitor. For multiple targets, u32Val should be bitwise OR-ed. For example, 0x10000003 will set validation targets COLOR_SPACE2_DCI_P3 and COLOR_SPACE2_BT_709

API
MONITOR_CODE SetCalibrationTarget (UWORD32 u32Val)

Params

u32Val

```
typedef enum color_space
{
    /* Pre-UP2720Q */
    COLOR_SPACE_ADOBE_RGB        = 0x00000001,
    COLOR_SPACE_SRGB             = 0x00000002,
    COLOR_SPACE_REC_709          = 0x00000004,
    COLOR_SPACE_DCI_P3          = 0x00000008,
    COLOR_SPACE_CAL_1           = 0x00000010,
    COLOR_SPACE_CAL_2           = 0x00000020,
    COLOR_SPACE_REC_2020        = 0x00000040,

    /* UP2720Q */
    COLOR_SPACE2_DCI_P3         = 0x10000001,
    COLOR_SPACE2_BT_709         = 0x10000002,
    COLOR_SPACE2_BT_2020        = 0x10000004,
    COLOR_SPACE2_SRGB           = 0x10000008,
    COLOR_SPACE2_ADOBE_RGB_D65  = 0x10000010,
    COLOR_SPACE2_ADOBE_RGB_D50  = 0x10000020,
    COLOR_SPACE2_NATIVE         = 0x10000040,
    COLOR_SPACE2_CUSTOM_1       = 0x10000080,
    COLOR_SPACE2_CUSTOM_2       = 0x10000100,
    COLOR_SPACE2_CUSTOM_3       = 0x10000200,
    COLOR_SPACE2_CAL_1          = 0x10000400,
    COLOR_SPACE2_CAL_2          = 0x10000800
}
COLOR_SPACE;
```

GetCalibrationSpeed

Return calibration speed value

API

MONITOR_CODE GetCalibrationSpeed(BYTE *pu8Val)

Params

*pu8Val	Pointer to return calibration speed value

Return

pu8Val

```
typedef enum calibration_speed
{
    CALIBRATION_SPEED_EXPRESS = 1, //Express
    CALIBRATION_SPEED_DETAIL  = 2  //Comprehensive
}
CALIBRATION_SPEED;
```

SetCalibrationSpeed

Set calibration speed value

API

MONITOR_CODE SetCalibrationSpeed(BYTE u8Val)

Params

u8Val

```
typedef enum calibration_speed
{
    CALIBRATION_SPEED_EXPRESS = 1, //Express
    CALIBRATION_SPEED_DETAIL  = 2  //Comprehensive
}
CALIBRATION_SPEED;
```

GetCalibrationWarmUp

Return calibration warm up value

API
MONITOR_CODE GetCalibrationWarmUp (BYTE *pu8Val)

Params

*pu8Val	Pointer to return calibration warm up value

Return

pu8Val	calibration warm up value

SetCalibrationWarmUp

Set calibration warm up value

API
MONITOR_CODE SetCalibrationWarmUp(BYTE u8Val)

Params

u8Val	calibration warm up value

GetColorimeterProfile

Return colorimeter profile value

API
MONITOR_CODE GetColorimeterProfile(BYTE *pu8Val)

Params

*pu8Val	Pointer to return colorimeter profile value

Return

pu8Val

```
typedef enum colorimeter_profile
{
    COLORIMETER_PROFILE_BUILT_IN = 1,
    COLORIMETER_PROFILE_CORRELATED = 2
}
COLORIMETER_PROFILE;
```

SetColorimeterProfile

Set colorimeter profile value

API
MONITOR_CODE SetColorimeterProfile(BYTE u8Val)

Params

u8Val

```
typedef enum colorimeter_profile
{
    COLORIMETER_PROFILE_BUILT_IN = 1,
    COLORIMETER_PROFILE_CORRELATED = 2
}
COLORIMETER_PROFILE;
```

GetCalibrationResult

Return calibration result for the particular color space mode

API

MONITOR_CODE GetCalibrationResult(UWORD32 u32ColorSpaceMode, CalibrationResultStructType *pData)

Params

u32ColorSpaceMode

```
typedef enum color_space_mode
{
    /* UP2720Q */
    COLOR_SPACE_MODE_DCI_P3             = 0x00000000,
    COLOR_SPACE_MODE_BT_709             = 0x00000001,
    COLOR_SPACE_MODE_BT_2020            = 0x00000002,
    COLOR_SPACE_MODE_SRGB               = 0x00000003,
    COLOR_SPACE_MODE_ADOBE_RGB_D65      = 0x00000004,
    COLOR_SPACE_MODE_ADOBE_RGB_D50      = 0x00000005,
    COLOR_SPACE_MODE_CAL_1              = 0x00000006,
    COLOR_SPACE_MODE_CAL_2              = 0x00000007,
    COLOR_SPACE_MODE_DCI_P3_UC          = 0x00000008,
    COLOR_SPACE_MODE_BT_709_UC          = 0x00000009,
    COLOR_SPACE_MODE_BT_2020_UC         = 0x0000000A,
    COLOR_SPACE_MODE_SRGB_UC            = 0x0000000B,
    COLOR_SPACE_MODE_ADOBE_RGB_D65_UC   = 0x0000000C,
    COLOR_SPACE_MODE_ADOBE_RGB_D50_UC   = 0x0000000D,
    COLOR_SPACE_MODE_CAL_1_UC           = 0x0000000E,
    COLOR_SPACE_MODE_CAL_2_UC           = 0x0000000F
}
COLOR_SPACE_MODE;
```

*pData — Pointer to return calibration result

Return

pData

```
typedef struct CalibrationResultStruct {
    UWORD32 ColorSpaceMode;           //refer to enum
    COLOR_SPACE_MODE;
    FLOAT RGBW[4][3];                 //4 patterns
    (X,Y,Z)
    FLOAT Gray[16][3];                //16 patterns
    (X,Y,Z)
    FLOAT Luminance;                  //Range:
    Luminance <= 400.
    BYTE GammaType;                  //refer to enum
    GAMMA_TYPE;
    FLOAT GammaValue;                 //Range: 10 <=
    GammaValue <= 26.
    BYTE stTargetCalibrationDate[5];
    //(mm/hh/DD/MM/YY) 5 bytes
    BYTE stActualCalibrationDate[5];
    //(mm/hh/DD/MM/YY) 5 bytes
    UWORD32 ColorTemp;                //Range: 2700 <=
    ColorTemp <= 10000.
    UWORD16 reserved;
}
CalibrationResultStructType;
```

StartCalibration

Start calibration

API

MONITOR_CODE StartCalibration(void)

Params

-

GetValidationTarget

Return validation targets as set in the monitor

API

MONITOR_CODE GetValidationTarget(UWORD32 *pu32Val)

Params

*pu32Val	Pointer to return validation target value

Return

pu32Val

```
typedef enum color_space
{
    /* Pre-UP2720Q */
    COLOR_SPACE_ADOBE_RGB        = 0x00000001,
    COLOR_SPACE_SRGB             = 0x00000002,
    COLOR_SPACE_REC_709          = 0x00000004,
    COLOR_SPACE_DCI_P3           = 0x00000008,
    COLOR_SPACE_CAL_1            = 0x00000010,
    COLOR_SPACE_CAL_2            = 0x00000020,
    COLOR_SPACE_REC_2020         = 0x00000040,

    /* UP2720Q */
    COLOR_SPACE2_DCI_P3          = 0x10000001,
    COLOR_SPACE2_BT_709          = 0x10000002,
    COLOR_SPACE2_BT_2020         = 0x10000004,
    COLOR_SPACE2_SRGB            = 0x10000008,
    COLOR_SPACE2_ADOBE_RGB_D65   = 0x10000010,
    COLOR_SPACE2_ADOBE_RGB_D50   = 0x10000020,
    COLOR_SPACE2_NATIVE          = 0x10000040,
    COLOR_SPACE2_CUSTOM_1        = 0x10000080,
    COLOR_SPACE2_CUSTOM_2        = 0x10000100,
    COLOR_SPACE2_CUSTOM_3        = 0x10000200,
    COLOR_SPACE2_CAL_1           = 0x10000400,
    COLOR_SPACE2_CAL_2           = 0x10000800
}
COLOR_SPACE;
```

SetValidationTarget

Set validation targets. For multiple targets, u32Val should be bitwise OR-ed. For example, 0x10000003 will set validation targets COLOR_SPACE2_DCI_P3 and COLOR_SPACE2_BT_709

API

MONITOR_CODE SetValidationTarget(UWORD32 u32Val)

Params

u32Val

```
typedef enum color_space
{
    /* Pre-UP2720Q */
    COLOR_SPACE_ADOBE_RGB        = 0x00000001,
    COLOR_SPACE_SRGB             = 0x00000002,
    COLOR_SPACE_REC_709          = 0x00000004,
    COLOR_SPACE_DCI_P3           = 0x00000008,
    COLOR_SPACE_CAL_1            = 0x00000010,
    COLOR_SPACE_CAL_2            = 0x00000020,
    COLOR_SPACE_REC_2020         = 0x00000040,

    /* UP2720Q */
    COLOR_SPACE2_DCI_P3          = 0x10000001,
    COLOR_SPACE2_BT_709          = 0x10000002,
    COLOR_SPACE2_BT_2020         = 0x10000004,
    COLOR_SPACE2_SRGB            = 0x10000008,
    COLOR_SPACE2_ADOBE_RGB_D65   = 0x10000010,
    COLOR_SPACE2_ADOBE_RGB_D50   = 0x10000020,
    COLOR_SPACE2_NATIVE          = 0x10000040,
    COLOR_SPACE2_CUSTOM_1        = 0x10000080,
    COLOR_SPACE2_CUSTOM_2        = 0x10000100,
    COLOR_SPACE2_CUSTOM_3        = 0x10000200,
    COLOR_SPACE2_CAL_1           = 0x10000400,
    COLOR_SPACE2_CAL_2           = 0x10000800
}
COLOR_SPACE;
```

GetAutoCalibrate

Return if auto calibrate is on/off.

API

MONITOR_CODE GetAutoCalibrate(BYTE *pu8Val)

Params

*pu8Val	Pointer to return auto calibrate value

Return

pu8Val	Auto calibrate value

SetAutoCalibrate

On /Off auto calibrate

API

MONITOR_CODE SetAutoCalibrate(BYTE u8Val)

Params

u8Val	Auto calibrate value

GetValidationPattern

Return validation pattern value

API

MONITOR_CODE GetValidationPattern(BYTE *pu8Val)

Params

*pu8Val	Point to return validation pattern value

Return

pu8Val

```
typedef enum validation_pattern
{
    VALIDATION_PATTERN_BASIC_RGB = 1,
    VALIDATION_PATTERN_LCD_COLOR_CHECKER = 2
}
VALIDATION_PATTERN;
```

SetValidationPattern

Set validation pattern value

API

MONITOR_CODE SetValidationPattern(BYTE u8Val)

Params

u8Val

```
typedef enum validation_pattern
{
    VALIDATION_PATTERN_BASIC_RGB = 1,
    VALIDATION_PATTERN_LCD_COLOR_CHECKER = 2
}
VALIDATION_PATTERN;
```

GetValidationResult

Return validation result for the particular color space mode

API

MONITOR_CODE GetValidationResult(UWORD32 u32ColorSpaceMode, ValidationResultStructType *pData)

Params

u32ColorSpaceMode

```
typedef enum color_space
{
    COLOR_SPACE_ADOBE_RGB       = 0x00000001,
    COLOR_SPACE_SRGB            = 0x00000002,
    COLOR_SPACE_REC_709         = 0x00000004,
    COLOR_SPACE_DCI_P3          = 0x00000008,
    COLOR_SPACE_CAL_1           = 0x00000010,
    COLOR_SPACE_CAL_2           = 0x00000020,
    COLOR_SPACE_REC_2020        = 0x00000040,

    /* UP2720Q */
    COLOR_SPACE2_DCI_P3         = 0x10000001,
    COLOR_SPACE2_BT_709         = 0x10000002,
    COLOR_SPACE2_BT_2020        = 0x10000004,
    COLOR_SPACE2_SRGB           = 0x10000008,
    COLOR_SPACE2_ADOBE_RGB_D65  = 0x10000010,
    COLOR_SPACE2_ADOBE_RGB_D50  = 0x10000020,
    COLOR_SPACE2_NATIVE         = 0x10000040,
    COLOR_SPACE2_CUSTOM_1       = 0x10000080,
    COLOR_SPACE2_CUSTOM_2       = 0x10000100,
    COLOR_SPACE2_CUSTOM_3       = 0x10000200,
    COLOR_SPACE2_CAL_1          = 0x10000400,
    COLOR_SPACE2_CAL_2          = 0x10000800
}
COLOR_SPACE;
```

*pData — Pointer to return validation result

Return

pData

```
typedef struct ValidationResultStruct {
    UWORD32 ColorSpaceMode;
    //refer to enum COLOR_SPACE_MODE;
    BYTE MeasureDataReady;
    BYTE Gamut;  //0x00: Native, 0x01: AdobeRGB, 0x02:
    sRGB, 0x03:DCI-P3, 0x06: REC709, 0x07: REC2020.
    BYTE GammaType;  //refer to enum GAMMA_TYPE;
    FLOAT GammaValue;
    //Range: 10 <= GammaValue <= 26.
    double Target_XYZ[41][3];     //Double X, Y, Z
    double Target_Lab[41][3];     //Double L, a, b
    double Measured_XYZ[49][3];   //Double X, Y, Z
    double Measured_Lab[41][3];   //Double L, a, b
    UWORD16 MeasuredXYZChecksum;
    FLOAT VerifiedGammaValue;
    FLOAT VerifiedColorTemp;
    FLOAT VerifiedGamutCoordinate[3][3];
    //FLOAT X, Y, Z
    double DeltaE76[41];
    double DeltaH94[41];
    double DeltaE94[41];
    double DeltaH2K[41];
    double DeltaE2K[41];
    BYTE stTargetValidationDate[5];
    //(mm/hh/DD/MM/YY) 5 bytes
    BYTE stActualValidationDate[5];
    //(mm/hh/DD/MM/YY) 5 bytes
    UWORD16 reserved;
}
ValidationResultStructType;
```

StartValidation

Start validation.

API
MONITOR_CODE StartValidation(void)

Params
-

GetCalibrationModulePowerState

Return if calibration module power state is on / off.

API
MONITOR_CODE GetCalibrationModulePowerState(BYTE *pu8Val)

Params

*pu8Val	Pointer to return calibration module power state value

Return

pu8Val	Calibration module power state

SetCalibrationModulePowerState

On / Off calibration module power.

API
MONITOR_CODE SetCalibrationModulePowerState(BYTE u8Val)

Params

u8Val	Calibration module power state

GetCalibrationValidationProgress

Return calibration validation progress.

API

MONITOR_CODE GetCalibrationValidationProgress(BYTE *pu8Val)

Params

*pu8Val	Pointer to return calibration validation progress value

Return

pu8Val	Calibration validation progress 0 Not in Calibration, Validation or setCorrelatedProfile 1 Calibration, Validation or setCorrelatedProfile in progress

AbortCalibrationValidation

Abort calibration, validation.

API

MONITOR_CODE AbortCalibrationValidation(void)

Params

-

GetCalibrationTargetInfo

Returns the calibration target info for the color space as specified in the structure

API

MONITOR_CODE GetCalibrationTargetInfo(CalibrationTargetInfoStructType *pData)

Params

* pData	Pointer to return calibration target info data

Return

pData	Set pData->ColorSpace to retrieve info of the particular color space

```
typedef struct CalibrationTargetInfoStruct {
    UWORD32 ColorSpace; //refer to enum COLOR_SPACE
    FLOAT Coordinate_R[2];    //(x,y) : 8bytes
    FLOAT Coordinate_G[2];    //(x,y) : 8bytes
    FLOAT Coordinate_B[2];    //(x,y) : 8bytes
    FLOAT Coordinate_W[2];    //(x,y) : 8bytes
    BYTE GammaValue;          //0x10-0x1A: 1.6-2.6,
    0x20:bt1886, 0x21:sRGB, 0x22:EPD, 0x24:EBU
    UWORD16 Luminance;
    BYTE UniformityStatus;    //0: OFF, 1: ON
}
CalibrationTargetInfoStructType;
```

SetCalibrationTargetInfo

Set calibration target info for the specified color space.

API

MONITOR_CODE SetCalibrationTargetInfo(CalibrationTargetInfoStructType *pData)

Params

u8Val

```
typedef struct CalibrationTargetInfoStruct {
    UWORD32 ColorSpace; //refer to enum COLOR_SPACE
    FLOAT Coordinate_R[2];   //(x,y) : 8bytes
    FLOAT Coordinate_G[2];   //(x,y) : 8bytes
    FLOAT Coordinate_B[2];   //(x,y) : 8bytes
    FLOAT Coordinate_W[2];   //(x,y) : 8bytes
    BYTE GammaValue;         //0x10-0x1A: 1.6-2.6,
    0x20:bt1886, 0x21:sRGB, 0x22:EPD, 0x24:EBU
    UWORD16 Luminance;
    BYTE UniformityStatus;   //0: OFF, 1: ON
}
CalibrationTargetInfoStructType;
```

Scheduler

GetCalValScheduler

Return if Calibration and Validation scheduler value.

API

MONITOR_CODE GetCalValScheduler(BYTE *pu8Val)

Params

*pu8Val	Pointer to return scheduler value

Return

pu8Val

```
typedef enum calvalscheduler
{
    CALVALSCHEDULER_OFF              = 0x00,
    CALVALSCHEDULER_CALIBRATION_ONLY = 0x01,
    CALVALSCHEDULER_VALIDATION_ONLY  = 0x02
}
CALVALSCHEDULER;
```

SetCalValScheduler

Set Calibration and Validation scheduler value.

API

MONITOR_CODE SetCalValScheduler(BYTE u8Val)

Params

u8Val

```
typedef enum calvalscheduler
{
    CALVALSCHEDULER_OFF              = 0x00,
    CALVALSCHEDULER_CALIBRATION_ONLY = 0x01,
    CALVALSCHEDULER_VALIDATION_ONLY  = 0x02
}
CALVALSCHEDULER;
```

GetCalValSchedule

Return Calibration and Validation schedule.

API

MONITOR_CODE GetCalValSchedule(BYTE *pu8Type, UWORD32 *pu32UsageQuarter, UWORD32 *pu32Week, UWORD32 *pu32Day, BYTE *pu8Hr, BYTE *pu8Min)

Params

*pu8Type	Pointer to return schedule type
*pu32UsageQuarter	Pointer to return schedule usage (pu8Type=1) *or* quarter
*pu32Week	Pointer to return schedule week
*pu32Day	Pointer to return schedule day
*pu8Hr	Pointer to return schedule hour

*pu8Min Pointer to return schedule minute

Return

pu8Type
```
typedef enum calvalschedule_type
{
    CALVALSCHEDULE_TYPE_BACKLIGHT_HRS = 0x01,
    CALVALSCHEDULE_TYPE_QUARTERLY     = 0x02,
    CALVALSCHEDULE_TYPE_MONTHLY       = 0x03,
    CALVALSCHEDULE_TYPE_WEEKLY        = 0x04,
    CALVALSCHEDULE_TYPE_DAILY         = 0x05
}
CALVALSCHEDULE_TYPE;
```
pu32UsageQuarter
```
typedef enum calvalschedule_quarter
{
    CALVALSCHEDULE_QUARTER_JAN = 0x00000001,
    /* Jan-Apr-Jul-Oct */
    CALVALSCHEDULE_QUARTER_FEB = 0x00000002,
    /* Feb-May-Aug-Nov */
    CALVALSCHEDULE_QUARTER_MAR = 0x00000003
    /* Mar-Jun-Sep-Dec */
}
CALVALSCHEDULE_QUARTER;
```
pu32Week
```
typedef enum calvalschedule_week
{
    CALVALSCHEDULE_WEEK_1      = 0x00000010,
    CALVALSCHEDULE_WEEK_2      = 0x00000020,
    CALVALSCHEDULE_WEEK_3      = 0x00000030,
    CALVALSCHEDULE_WEEK_4      = 0x00000040,
    CALVALSCHEDULE_WEEK_5      = 0x00000050 //unused
}
CALVALSCHEDULE_WEEK;
```
pu32Day
```
typedef enum calvalschedule_day
{
    CALVALSCHEDULE_DAY_MON     = 0x00000100,
    CALVALSCHEDULE_DAY_TUE     = 0x00000200,
    CALVALSCHEDULE_DAY_WED     = 0x00000300,
    CALVALSCHEDULE_DAY_THU     = 0x00000400,
    CALVALSCHEDULE_DAY_FRI     = 0x00000500,
    CALVALSCHEDULE_DAY_SAT     = 0x00000600,
    CALVALSCHEDULE_DAY_SUN     = 0x00000700,
    CALVALSCHEDULE_DAY_DAILY   = 0x00000800,
    CALVALSCHEDULE_DAY_WEEKDAY = 0x00000900
}
CALVALSCHEDULE_DAY;
```
pu8Hr hour (0-23)

pu8Min minute (0-59)

SetCalValSchedule

Set Calibration and Validation schedule.

API

MONITOR_CODE SetCalValSchedule(BYTE pu8Type, UWORD32 pu32UsageQuarter, UWORD32 pu32Week, UWORD32 pu32Day, BYTE pu8Hr, BYTE pu8Min)

Params

pu8Type
```
typedef enum calvalschedule_type
{
    CALVALSCHEDULE_TYPE_BACKLIGHT_HRS = 0x01,
    CALVALSCHEDULE_TYPE_QUARTERLY     = 0x02,
    CALVALSCHEDULE_TYPE_MONTHLY       = 0x03,
    CALVALSCHEDULE_TYPE_WEEKLY        = 0x04,
    CALVALSCHEDULE_TYPE_DAILY         = 0x05
}
CALVALSCHEDULE_TYPE;
```

pu32UsageQuarter

```
typedef enum calvalschedule_quarter
{
    CALVALSCHEDULE_QUARTER_JAN = 0x00000001,
    /* Jan-Apr-Jul-Oct */
    CALVALSCHEDULE_QUARTER_FEB = 0x00000002,
    /* Feb-May-Aug-Nov */
    CALVALSCHEDULE_QUARTER_MAR = 0x00000003
    /* Mar-Jun-Sep-Dec */
}
CALVALSCHEDULE_QUARTER;
```

pu32Week

```
typedef enum calvalschedule_week
{
    CALVALSCHEDULE_WEEK_1      = 0x00000010,
    CALVALSCHEDULE_WEEK_2      = 0x00000020,
    CALVALSCHEDULE_WEEK_3      = 0x00000030,
    CALVALSCHEDULE_WEEK_4      = 0x00000040,
    CALVALSCHEDULE_WEEK_5      = 0x00000050 //unused
}
CALVALSCHEDULE_WEEK;
```

pu32Day

```
typedef enum calvalschedule_day
{
    CALVALSCHEDULE_DAY_MON     = 0x00000100,
    CALVALSCHEDULE_DAY_TUE     = 0x00000200,
    CALVALSCHEDULE_DAY_WED     = 0x00000300,
    CALVALSCHEDULE_DAY_THU     = 0x00000400,
    CALVALSCHEDULE_DAY_FRI     = 0x00000500,
    CALVALSCHEDULE_DAY_SAT     = 0x00000600,
    CALVALSCHEDULE_DAY_SUN     = 0x00000700,
    CALVALSCHEDULE_DAY_DAILY   = 0x00000800,
    CALVALSCHEDULE_DAY_WEEKDAY = 0x00000900
}
CALVALSCHEDULE_DAY;
```

pu8Hr	hour (0-23)
pu8Min	minute (0-59)

GetCalValOpMode

Return Calibration and Validation operation mode

API

MONITOR_CODE GetCalValOpMode(BYTE *pu8Val)

Params

*pu8Val	Point to return operation mode value

Return

pu8Val

```
typedef enum calvalschedule_op_mode
{
    CALVALSCHEDULER_OP_MODE_PROMPT = 1,
    CALVALSCHEDULER_OP_MODE_SLEEP = 2
}
CALVALSCHEDULER_OP_MODE;
```

SetCalValOpMode

Set Calibration and Validation operation mode

API

MONITOR_CODE SetCalValOpMode(BYTE u8Val)

Params

u8Val	`typedef enum calvalschedule_op_mode { CALVALSCHEDULER_OP_MODE_PROMPT = 1, CALVALSCHEDULER_OP_MODE_SLEEP = 2 } CALVALSCHEDULER_OP_MODE;`

```
typedef enum calvalschedule_op_mode
{
    CALVALSCHEDULER_OP_MODE_PROMPT = 1,
    CALVALSCHEDULER_OP_MODE_SLEEP = 2
}
CALVALSCHEDULER_OP_MODE;
```

Example Flow

Application

Example initialization and connecting to a monitor

1. Initialize the SDK: **Initialize()**
2. Get connected monitors
    a. Use **GetAvailableMonitors** to just get a count, *OR*
    b. Use **GetAvailableMonitorsDetail** to get count and the associated array of monitor model name
3. Optionally, show index on the monitors if count more than 1: **IdentifyMonitor()**
   Note that index shown will be (index+1). So first monitor (index 0) will be shown as Monitor 1.
4. Connect to monitor using index: **ConnectMonitor(index)** where index is 0 to (count-1) returned in step 2.
5. Perform your application processes…
6. Disconnect monitor: **DisconnectMonitor()**
7. Shutdown the SDK: **Shutdown()**

Note that SDK can only connect to 1 monitor at any single point of time