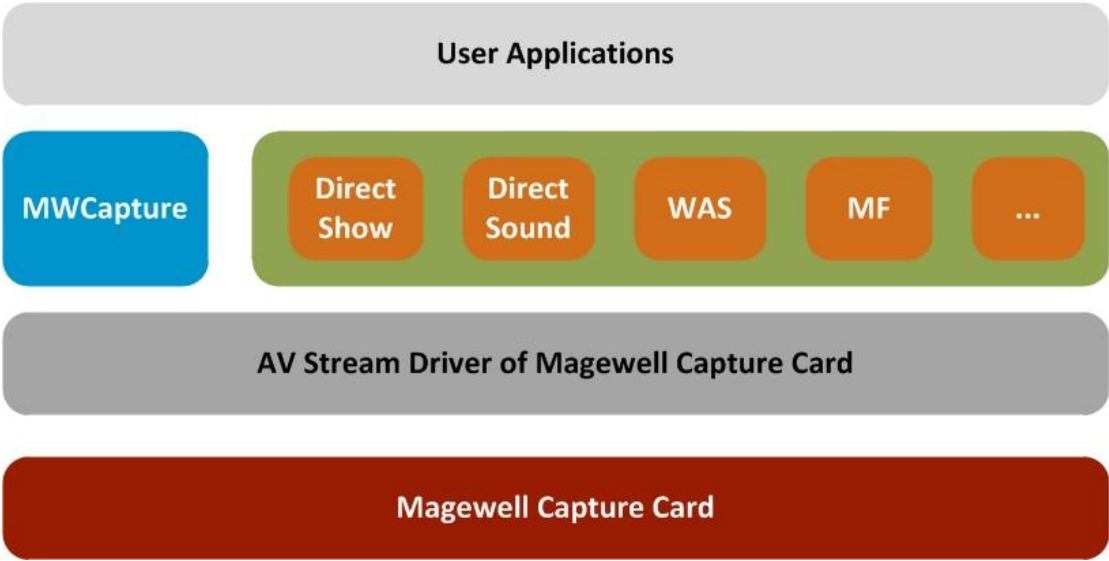


Pro Capture Series Card Software Development Manual

1 Overview.....	2
2 Type Definitions	3
2.1 Enumerated Types.....	3
2.2 Data Structures.....	17
3 Functions.....	45
3.1 DirectShow Expansion Interface Functions	45
3.2 MWCapture Interface Functions	80
4 Contact Us.....	136

1 Overview

SDK is mainly used for developing application software with Pro Capture Series Card. SDK directly access Pro Capture Series card through MWCapture interfaces. By using this SDK, users can get all features of Pro Capture Series Card and reach the best performance and better flexibility. DirectShow extension interfaces are also provided for DirectShow users to access extended features.



This document mainly introduces the specific functions and type definitions of software development kit. The introduction of features refer to “Software Development Guide” .

Type and Functions Index

DirectShow ExtensionInterface	Type and Property Definition refer to Chapter 2 andInterface functions refer to Section 1, Chapter 3.
MWCapture Functions	Type Definitions refer to Chapter 2 and Functions refer to Section 2, Chapter 3.

2 Type Definitions

DirectShow Extension Interface and API Functions use the same type definition

2.1 Enumerated Types

2.1.1 MW_FAMILY_ID Enumerated

Defines product family for capture device.

Syntax

```
typedef enum _MW_FAMILY_ID {  
    MW_FAMILY_ID_PRO_CAPTURE           = 0x00,  
    MW_FAMILY_ID_VALUE_CAPTURE         = 0x01,  
    MW_FAMILY_ID_USB_CAPTURE           = 0x02  
} MW_FAMILY_ID;
```

Requirement

Header | LibMWCapture\MWCommon.h

Constants

MW_FAMILY_ID_PRO_CAPTURE

The capture device use PCI-e slot to connect with computer.

MW_FAMILY_ID_VALUE_CAPTURE

The capture device use PCI-e slot to connect with computer. They do not use onboard buffer.

MW_FAMILY_ID_USB_CAPTURE

The capture device use USB port to connect with computer.

2.1.2 MWCAP_PRODUCT_ID Enumerated

Defines product type for capture device. Different types of devices have different input/output interfaces.

Syntax

```
typedef enum _MWCAP_PRODUCT_ID {  
    MWCAP_PRODUCT_ID_PRO_CAPTURE_AIO           = 0x00000102,  
    MWCAP_PRODUCT_ID_PRO_CAPTURE_DVI           = 0x00000103,  
    MWCAP_PRODUCT_ID_PRO_CAPTURE_HDMI           = 0x00000104,  
    MWCAP_PRODUCT_ID_PRO_CAPTURE_SDI           = 0x00000105,  
    MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_SDI       = 0x00000106,  
    MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_DVI       = 0x00000107,  
    MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_HDMI      = 0x00000108,  
}
```

MWCAP_PRODUCT_ID_PRO_CAPTURE_QUAD_SDI	= 0x00000109,
MWCAP_PRODUCT_ID_PRO_CAPTURE_QUAD_HDMI	= 0x00000110,
MWCAP_PRODUCT_ID_PRO_CAPTURE_MINI_HDMI	= 0x00000111,
MWCAP_PRODUCT_ID_PRO_CAPTURE_HDMI_4K	= 0x00000112,
MWCAP_PRODUCT_ID_PRO_CAPTURE_MINI_SDI	= 0x00000113
} MWCAP_PRODUCT_ID;	

Requirement

Header | LibMWCapture\MWCAptureExtension.h

Constants

MWCAP_PRODUCT_ID_PRO_CAPTURE_AIO

The capture device has PCI-e slot.

MWCAP_PRODUCT_ID_PRO_CAPTURE_DVI

The capture device has PCI-e host slot and DVI input slot.

MWCAP_PRODUCT_ID_PRO_CAPTURE_HDMI

The capture device has PCI-e host slot and HDMI input slot.

MWCAP_PRODUCT_ID_PRO_CAPTURE_SDI

The capture device has PCI-e host slot and SDI input slot.

MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_SDI

The capture device has PCI-e host slot and two SDI input slots.

MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_DVI

The capture device has PCI-e host slot and two DVI input slots.

MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_HDMI

The capture device has PCI-e host slot and two HDMI input slots.

MWCAP_PRODUCT_ID_PRO_CAPTURE_QUAD_SDI

The capture device has PCI-e host slot and four SDI input slots.

MWCAP_PRODUCT_ID_PRO_CAPTURE_QUAD_HDMI

The capture device has PCI-e host slot and four HDMI input slots.

MWCAP_PRODUCT_ID_PRO_CAPTURE_MINI_HDMI

The capture device has PCI-e host slot and mini HDMI input slot.

MWCAP_PRODUCT_ID_PRO_CAPTURE_HDMI_4K

The capture device has PCI-e host slot and 4K HDMI input slot.

MWCAP_PRODUCT_ID_PRO_CAPTURE_MINI_SDI

The capture device has PCI-e host slot and mini SDI input slot.

2.1.3 MWCAP_VIDEO_INPUT_TYPE Enumerated

Defines input type mask for video channel.

Syntax

typedef enum _MWCAP_VIDEO_INPUT_TYPE {	
MWCAP_VIDEO_INPUT_TYPE_NONE	= 0x00,
MWCAP_VIDEO_INPUT_TYPE_HDMI	= 0x01,

MWCAP_VIDEO_INPUT_TYPE_VGA	= 0x02,
MWCAP_VIDEO_INPUT_TYPE_SDI	= 0x04,
MWCAP_VIDEO_INPUT_TYPE_COMPONENT	= 0x08,
MWCAP_VIDEO_INPUT_TYPE_CVBS	= 0x10,
MWCAP_VIDEO_INPUT_TYPE_YC	= 0x20
} MWCAP_VIDEO_INPUT_TYPE;	

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_INPUT_TYPE_NONE

No input signal type.

MWCAP_VIDEO_INPUT_TYPE_HDMI

Input interface of HDMI signal

MWCAP_VIDEO_INPUT_TYPE_VGA

Input interface of VGA signal.

MWCAP_VIDEO_INPUT_TYPE_SDI

Input interface of SDI signal.

MWCAP_VIDEO_INPUT_TYPE_COMPONENT

Input interface of Component signal

MWCAP_VIDEO_INPUT_TYPE_CVBS

Input interface of CVBS signal.

MWCAP_VIDEO_INPUT_TYPE_YC

Input interface of YC signal

2.1.4 MWCAP_AUDIO_INPUT_TYPE Enumerated

Defines input type mask for audio channel

Syntax

typedef enum _MWCAP_AUDIO_INPUT_TYPE {	
MWCAP_AUDIO_INPUT_TYPE_NONE	= 0x00,
MWCAP_AUDIO_INPUT_TYPE_HDMI	= 0x01,
MWCAP_AUDIO_INPUT_TYPE_SDI	= 0x02,
MWCAP_AUDIO_INPUT_TYPE_LINE_IN	= 0x04,
MWCAP_AUDIO_INPUT_TYPE_MIC_IN	= 0x08
} MWCAP_AUDIO_INPUT_TYPE;	

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_AUDIO_INPUT_TYPE_NONE

No input signal type

MWCAP_AUDIO_INPUT_TYPE_HDMI

Input interface of HDMI signal

MWCAP_AUDIO_INPUT_TYPE_SDI

Input interface of SDI signal

MWCAP_AUDIO_INPUT_TYPE_LINE_IN

Input interface of line input signal

MWCAP_AUDIO_INPUT_TYPE_MIC_IN

Input interface of microphoneinput signal

2.1.5 MWCAP_PCIE_LINK_TYPE Enumerated

Defines audio input slot type for capture device.

Syntax

```
typedef enum _MWCAP_PCIE_LINK_TYPE {  
    MWCAP_PCIE_LINK_GEN_1                = 0x01,  
    MWCAP_PCIE_LINK_GEN_2                = 0x02,  
    MWCAP_PCIE_LINK_GEN_3                = 0x04,  
    MWCAP_PCIE_LINK_GEN_4                = 0x08  
} MWCAP_PCIE_LINK_TYPE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_PCIE_LINK_GEN_1

PCI-e 1.0 transmission standard.

MWCAP_PCIE_LINK_GEN_2

PCI-e 2.0 transmission standard.

MWCAP_PCIE_LINK_GEN_3

PCI-e 3.0 transmission standard.

MWCAP_PCIE_LINK_GEN_4

PCI-e 4.0 transmission standard.

2.1.6 MWCAP_VIDEO_TIMING_TYPE Enumerated

Defines audio input slot type for capture device.

Syntax

```
typedef enum _MWCAP_VIDEO_TIMING_TYPE {  
    MWCAP_VIDEO_TIMING_NONE                = 0x00000000,  
    MWCAP_VIDEO_TIMING_LEGACY              = 0x00000001,  
    MWCAP_VIDEO_TIMING_DMT                 = 0x00000002,  
}
```

MWCAP_VIDEO_TIMING_CEA	= 0x00000004,
MWCAP_VIDEO_TIMING_GTF	= 0x00000008,
MWCAP_VIDEO_TIMING_CVT	= 0x00000010,
MWCAP_VIDEO_TIMING_CVT_RB	= 0x00000020,
MWCAP_VIDEO_TIMING_FAILSAFE	= 0x00002000
} MWCAP_VIDEO_TIMING_TYPE;	

Requirement

Header | LibMWCapture\MWCAptureExtension.h

Constants

MWCAP_VIDEO_TIMING_NONE

Video channel has no timing.

MWCAP_VIDEO_TIMING_LEGACY

Video channel uses LEGACY timing type.

MWCAP_VIDEO_TIMING_DMT

Video channel uses LEGACY DMT type.

MWCAP_VIDEO_TIMING_CEA

Video channel uses CEA timing type.

MWCAP_VIDEO_TIMING_GTF

Video channel uses GTF timing type.

MWCAP_VIDEO_TIMING_CVT

Video channel uses CVT timing type.

MWCAP_VIDEO_TIMING_CVT_RB

Video channel uses CVT_RB timing type.

MWCAP_VIDEO_TIMING_FAILSAFE

Video channel uses FAILSAFE timing type.

2.1.7 MWCAP_VIDEO_COLOR_FORMAT Enumerated

Defines audio input slot type for capture device.

Syntax

typedef enum _MWCAP_VIDEO_COLOR_FORMAT {	
MWCAP_VIDEO_COLOR_FORMAT_UNKNOWN	= 0x00,
MWCAP_VIDEO_COLOR_FORMAT_RGB	= 0x01,
MWCAP_VIDEO_COLOR_FORMAT_YUV601	= 0x02,
MWCAP_VIDEO_COLOR_FORMAT_YUV709	= 0x03,
MWCAP_VIDEO_COLOR_FORMAT_YUV2020	= 0x04,
MWCAP_VIDEO_COLOR_FORMAT_YUV2020C	= 0x05
} MWCAP_VIDEO_COLOR_FORMAT;	

Requirement

Header | LibMWCapture\MWCAptureExtension.h

Constants

MWCAP_VIDEO_COLOR_FORMAT_UNKNOWN

Unknown color format.

MWCAP_VIDEO_COLOR_FORMAT_RGB

RGB color format.

MWCAP_VIDEO_COLOR_FORMAT_YUV601

YUV601 color format.

MWCAP_VIDEO_COLOR_FORMAT_YUV709

YUV709 color format.

MWCAP_VIDEO_COLOR_FORMAT_YUV2020

YUV2020 color format.

MWCAP_VIDEO_COLOR_FORMAT_YUV2020C

YUV2020C color format.

2.1.8 MWCAP_VIDEO_QUANTIZATION_RANGE Enumerated

Defines video signal quantization range.

Syntax

```
typedef enum _MWCAP_VIDEO_QUANTIZATION_RANGE {  
    MWCAP_VIDEO_QUANTIZATION_UNKNOWN    = 0x00,  
    MWCAP_VIDEO_QUANTIZATION_FULL       = 0x01,  
    MWCAP_VIDEO_QUANTIZATION_LIMITED    = 0x02  
} MWCAP_VIDEO_QUANTIZATION_RANGE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_QUANTIZATION_UNKNOWN

Unknown video quantization range.

MWCAP_VIDEO_QUANTIZATION_FULL

Full video quantization range. In this range, black and white color in 8 bits ranges from 0 to 255.

MWCAP_VIDEO_QUANTIZATION_LIMITED

Limited video quantization range. In this range, black and white color in 8 bits ranges from 16 to 255.

2.1.9 MWCAP_VIDEO_SATURATION_RANGE Enumerated

Defines video signal saturation range.

Syntax


```
typedef enum _MWCAP_VIDEO_SATURATION_RANGE {
    MWCAP_VIDEO_SATURATION_UNKNOWN          = 0x00,
    MWCAP_VIDEO_SATURATION_FULL              = 0x01,
    MWCAP_VIDEO_SATURATION_LIMITED           = 0x02,
    MWCAP_VIDEO_SATURATION_EXTENDED_GAMUT    = 0x03
} MWCAP_VIDEO_SATURATION_RANGE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_SATURATION_UNKNOWN

Unknown video saturation range.

MWCAP_VIDEO_SATURATION_FULL

Full video saturation range. In this range , black and white color in 8 bits ranges from 0 to 255.

MWCAP_VIDEO_SATURATION_LIMITED

Limited video saturation range. In this range, black and white color in 8 bits ranges from 16 to 235.

MWCAP_VIDEO_SATURATION_EXTENDED_GAMUT

Extended video saturation range. In this range, black and white color in 8 bits ranges from 1 to 254.

2.1.10 MWCAP_VIDEO_FRAME_TYPE Enumerated

Defines video frame type.

Syntax

```
typedef enum _MWCAP_VIDEO_FRAME_TYPE {
    MWCAP_VIDEO_FRAME_2D                      = 0x00,
    MWCAP_VIDEO_FRAME_3D_TOP_AND_BOTTOM_FULL  = 0x01,
    MWCAP_VIDEO_FRAME_3D_TOP_AND_BOTTOM_HALF  = 0x02,
    MWCAP_VIDEO_FRAME_3D_SIDE_BY_SIDE_FULL    = 0x03,
    MWCAP_VIDEO_FRAME_3D_SIDE_BY_SIDE_HALF    = 0x04
} MWCAP_VIDEO_FRAME_TYPE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_FRAME_2D

2D video frame.

MWCAP_VIDEO_FRAME_3D_TOP_AND_BOTTOM_FULL

Full 3D video frame with top and bottom fields.

MWCAP_VIDEO_FRAME_3D_TOP_AND_BOTTOM_HALF

Half 3D video frame with top and bottom fields

MWCAP_VIDEO_FRAME_3D_SIDE_BY_SIDE_FULL

Full side-by-side 3D video frame.

MWCAP_VIDEO_FRAME_3D_SIDE_BY_SIDE_HALF

Half side-by-side 3D video frame.

2.1.11 MWCAP_VIDEO_DEINTERLACE_MODE Enumerated

Defines video signal De-interlace mode.

Syntax

```
typedef enum _MWCAP_VIDEO_DEINTERLACE_MODE {  
    MWCAP_VIDEO_DEINTERLACE_WEAVE          = 0x00,  
    MWCAP_VIDEO_DEINTERLACE_BLEND          = 0x01,  
    MWCAP_VIDEO_DEINTERLACE_TOP_FIELD      = 0x02,  
    MWCAP_VIDEO_DEINTERLACE_BOTTOM_FIELD   = 0x03  
} MWCAP_VIDEO_DEINTERLACE_MODE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_DEINTERLACE_WEAVE

De-interlace with weave mode.

MWCAP_VIDEO_DEINTERLACE_BLEND

De-interlace with blend mode..

MWCAP_VIDEO_DEINTERLACE_TOP_FIELD

De-interlace by only using top field.

MWCAP_VIDEO_DEINTERLACE_BOTTOM_FIELD

De-interlace by only using bottom field

2.1.12 MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODE Enumerated

Defines video signal aspect ratio convert mode.

Syntax

```
typedef enum _MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODE {  
    MWCAP_VIDEO_ASPECT_RATIO_IGNORE        = 0x00,  
    MWCAP_VIDEO_ASPECT_RATIO_CROPPING      = 0x01,  
    MWCAP_VIDEO_ASPECT_RATIO_PADDING       = 0x02  
} MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_ASPECT_RATIO_IGNORE

Ignore aspect ratio.

MWCAP_VIDEO_ASPECT_RATIO_CROPPING

Convert aspect ratio by cropping.

MWCAP_VIDEO_ASPECT_RATIO_PADDING

Convert aspect ratio by padding.

2.1.13 MWCAP_VIDEO_SYNC_TYPE Enumerated

Defines video signal synchronous mode.

Syntax

```
typedef enum _MWCAP_VIDEO_SYNC_TYPE {  
    VIDEO_SYNC_ALL                = 0x07,  
    VIDEO_SYNC_HS_VS              = 0x01,  
    VIDEO_SYNC_CS                 = 0x02,  
    VIDEO_SYNC_EMBEDDED          = 0x04  
} MWCAP_VIDEO_SYNC_TYPE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

VIDEO_SYNC_ALL

Ignore aspect ratio.

VIDEO_SYNC_HS_VS

Convert aspect ratio by cropping.

VIDEO_SYNC_CS

Convert aspect ratio by padding.

VIDEO_SYNC_EMBEDDED

Convert aspect ratio by padding.

2.1.14 MWCAP_LED_MODE Enumerated

Defines LED mode.

Syntax

```
typedef enum _MWCAP_LED_MODE {  
    MWCAP_LED_AUTO                = 0x00000000,  
    MWCAP_LED_OFF                 = 0x80000000,  
    MWCAP_LED_ON                  = 0x80000001,  
    MWCAP_LED_BLINK               = 0x80000002,  
}
```

MWCAP_LED_DBL_BLINK	= 0x80000003,
MWCAP_LED_BREATH	= 0x80000004
} MWCAP_LED_MODE;	

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_LED_AUTO

Auto mode.

MWCAP_LED_OFF

Keep LED off.

MWCAP_LED_ON

Keep LED on.

MWCAP_LED_BLINK

Keep LED blinking.

MWCAP_LED_DBL_BLINK

Keep LED blinking twice each time

MWCAP_LED_BREATH

Breath mode.

2.1.15 MWCAP_SD_VIDEO_STANDARD Enumerated

Defines used video standard.

Syntax

```
typedef enum _MWCAP_SD_VIDEO_STANDARD {
    MWCAP_SD_VIDEO_NONE,
    MWCAP_SD_VIDEO_NTSC_M,
    MWCAP_SD_VIDEO_NTSC_433,
    MWCAP_SD_VIDEO_PAL_M,
    MWCAP_SD_VIDEO_PAL_60,
    MWCAP_SD_VIDEO_PAL_COMBN,
    MWCAP_SD_VIDEO_PAL_BGHID,
    MWCAP_SD_VIDEO_SECAM,
    MWCAP_SD_VIDEO_SECAM_60
} MWCAP_SD_VIDEO_STANDARD;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_SD_VIDEO_NONE

None video standard

MWCAP_SD_VIDEO_NTSC_M

Use NTSC_M standard

MWCAP_SD_VIDEO_NTSC_433

Use NTSC_433 standard

MWCAP_SD_VIDEO_PAL_M

Use PAL_M standard

MWCAP_SD_VIDEO_PAL_60

Use PAL_60 standard

MWCAP_SD_VIDEO_PAL_COMBN

Use PAL_COMBN standard

MWCAP_SD_VIDEO_PAL_BGHID

Use PAL_BGHID standard

MWCAP_SD_VIDEO_SECAM

Use SECAM standard

MWCAP_SD_VIDEO_SECAM_60

Use SECAM_60standard

2.1.16 MWCAP_VIDEO_SIGNAL_STATE Enumerated

Defines input video signal state.

Syntax

```
typedef enum _MWCAP_VIDEO_SIGNAL_STATE {  
    MWCAP_VIDEO_SIGNAL_NONE,  
    MWCAP_VIDEO_SIGNAL_UNSUPPORTED,  
    MWCAP_VIDEO_SIGNAL_LOCKING,  
    MWCAP_VIDEO_SIGNAL_LOCKED  
} MWCAP_VIDEO_SIGNAL_STATE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_SIGNAL_STATE_NONE

No input video signal.

MWCAP_VIDEO_SIGNAL_UNSUPPORTED

Invalid video signal. Input signal is detected by the capture device, but cannot be locked and recognized.

MWCAP_VIDEO_SINGAL_STATE_LOCKING

Locking video signal.Video signal status valid but not locked yet

MWCAP_VIDEO_SINGAL_STATE_LOCKED

Video signal has been locked. Capture device can capture current input signal.

2.1.17 MWCAP_VIDEO_FRAME_STATE Enumerated

Defines video frame state

Syntax

```
typedef enum _MWCAP_VIDEO_FRAME_STATE {  
    MWCAP_VIDEO_FRAME_STATE_INITIAL,  
    MWCAP_VIDEO_FRAME_STATE_F0_BUFFERING,  
    MWCAP_VIDEO_FRAME_STATE_F1_BUFFERING,  
    MWCAP_VIDEO_FRAME_STATE_BUFFERED  
} MWCAP_VIDEO_FRAME_STATE;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_VIDEO_FRAME_STATE_INITIAL

Initial state.

MWCAP_VIDEO_FRAME_STATE_F0_BUFFERING

Field 0 is buffing

MWCAP_VIDEO_FRAME_STATE_F1_BUFFERING

Field 1 is buffing

MWCAP_VIDEO_FRAME_STATE_BUFFERED

Video frame has been buffered

2.1.18 MWCAP_HDMI_INFOFRAME_ID Enumerated

定义 HDMI 信号信息帧的 ID。

Defines info frame ID of HDMI signal

Syntax

```
typedef enum _MWCAP_HDMI_INFOFRAME_ID {  
    MWCAP_HDMI_INFOFRAME_ID_AVI,  
    MWCAP_HDMI_INFOFRAME_ID_AUDIO,  
    MWCAP_HDMI_INFOFRAME_ID_SPD,  
    MWCAP_HDMI_INFOFRAME_ID_MS,  
    MWCAP_HDMI_INFOFRAME_ID_VS,  
    MWCAP_HDMI_INFOFRAME_ID_ACP,  
    MWCAP_HDMI_INFOFRAME_ID_ISRC1,  
    MWCAP_HDMI_INFOFRAME_ID_ISRC2,  
    MWCAP_HDMI_INFOFRAME_ID_GAMUT,  
    MWCAP_HDMI_INFOFRAME_COUNT  
} MWCAP_HDMI_INFOFRAME_ID;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Constants

MWCAP_HDMI_INFOFRAME_ID_AVI,

AVI info frame.

MWCAP_HDMI_INFOFRAME_ID_AUDIO

Aduio info frame.

MWCAP_HDMI_INFOFRAME_ID_SPD

SPD info frame.

MWCAP_HDMI_INFOFRAME_ID_MS

MS info frame.

MWCAP_HDMI_INFOFRAME_ID_VS

VS info frame.

MWCAP_HDMI_INFOFRAME_ID_ACP

ACP info frame.

MWCAP_HDMI_INFOFRAME_ID_ISRC1

ISRC1 info frame.

MWCAP_HDMI_INFOFRAME_ID_ISRC2

ISRC2 info frame.

MWCAP_HDMI_INFOFRAME_ID_GAMUT

GAMUTinfo frame.

MWCAP_HDMI_INFOFRAME_ID_COUNT

Count of all info frame types.

2.1.19 MWCAP_HDMI_INFOFRAME_MASK Enumerated

Defines info frame mask of HDMI signal

Syntax

```
typedef enum _MWCAP_HDMI_INFOFRAME_MASK {  
    MWCAP_HDMI_INFOFRAME_MASK_AVI    =(1<<MWCAP_HDMI_INFOFRAME_ID_AVI),  
    MWCAP_HDMI_INFOFRAME_MASK_AUDIO =(1<< MWCAP_HDMI_INFOFRAME_ID_AUDIO),  
    MWCAP_HDMI_INFOFRAME_MASK_SPD    =(1<< MWCAP_HDMI_INFOFRAME_ID_SPD),  
    MWCAP_HDMI_INFOFRAME_MASK_MS     =(1<< MWCAP_HDMI_INFOFRAME_ID_MS),  
    MWCAP_HDMI_INFOFRAME_MASK_VS     =(1<< MWCAP_HDMI_INFOFRAME_ID_VS),  
    MWCAP_HDMI_INFOFRAME_MASK_ACP    =(1<< MWCAP_HDMI_INFOFRAME_ID_ACP),  
    MWCAP_HDMI_INFOFRAME_MASK_ISRC1  =(1<< MWCAP_HDMI_INFOFRAME_ID_ISRC1),  
    MWCAP_HDMI_INFOFRAME_MASK_ISRC2  =(1<< MWCAP_HDMI_INFOFRAME_ID_ISRC2),  
    MWCAP_HDMI_INFOFRAME_MASK_GAMUT=(1<< WCAP_HDMI_INFOFRAME_ID_GAMUT)  
} MWCAP_HDMI_INFOFRAME_MASK;
```

Requirement

Header | LibMWCapture\MWCAptureExtension.h

Constants

MWCAP_HDMI_INFOFRAME_MASK_AVI,
AVI info frame.

MWCAP_HDMI_INFOFRAME_MASK_AUDIO
Aduio info frame.

MWCAP_HDMI_INFOFRAME_MASK_SPD
SPD info frame.

MWCAP_HDMI_INFOFRAME_MASK_MS
MS info frame.

MWCAP_HDMI_INFOFRAME_MASK_VS
VS info frame.

MWCAP_HDMI_INFOFRAME_MASK_ACP
ACP info frame.

MWCAP_HDMI_INFOFRAME_MASK_ISRC1
ISRC1 info frame.

MWCAP_HDMI_INFOFRAME_MASK_ISRC2
ISRC2 info frame.

MWCAP_HDMI_INFOFRAME_MASK_GAMUT
GAMUTinfo frame.

2.2 Data Structures

2.2.1 MWCAP_CHANNEL_INFO structure

The structure describes channel information.

Syntax

```
typedef struct _MWCAP_CHANNEL_INFO {  
    WORD                wFamilyID;  
    WORD                wProductID;  
    CHAR                chHardwareVersion;  
    BYTE                byFirmwareID;  
    DWORD               dwFirmwareVersion;  
    DWORD               dwDriverVersion;  
    CHAR                szFamilyName[MW_FAMILY_NAME_LEN];  
    CHAR                szProductName[MW_PRODUCT_NAME_LEN];  
    CHAR                szFirmwareName[MW_FIRMWARE_NAME_LEN];  
    CHAR                szBoardSerialNo[MW_SERIAL_NO_LEN];  
    BYTE                byBoardIndex;  
    BYTE                byChannelIndex;  
} MWCAP_CHANNEL_INFO;
```

Requirement

Header | LibMWCapture\MWCaptureExtension.h

Members

wFamilyID

Receives product family of capture device that channel is belonged to.

wProductID

Receives product id of capture device that channel is belonged to.

chHardwareVersion

Receives hardware version of capture device that channel is belonged to.

byFirmwareID

Receives firmware ID of capture device that channel is belonged to.

dwFirmwareVersion

Receives firmware version of capture device that channel is belonged to.

dwDriverVersion

Receives driver version of capture device that channel is belonged to.

szProductFamilyName

Receives product family name of capture device that channel is belonged to.

szProductName

Receives product family name of capture device that channel is belonged to.

szFirmwareName

Receives firmware name of capture device that channel is belonged to.

szBoardSerialNo

Receives board serial NO of capture device that channel is belonged to.

byBoardIndex

Board index. The value of rotary switch on the board. The value always begin from 0.

byChannelIndex

Receives channel index. A capture device may have multiple channels that their index will begin from 0.

2.2.2 MWCAP_PRO_CAPTURE_INFO structure

The structure describes capture device using PCI-e interface information.

Syntax

```
typedef struct _MWCAP_PRO_CAPTURE_INFO {  
    BYTE byPCIBusID;  
    BYTE byPCIDevID;  
    BYTE byLinkType;  
    BYTE byLinkWidth;  
    BYTE byBoardIndex;  
    WORD wMaxPayloadSize;  
    WORD wMaxReadRequestSize;  
    DWORD cbTotalMemorySize;  
    DWORD cbFreeMemorySize;  
} MWCAP_PRO_CAPTURE_INFO;
```

Members

byPCIBusID

Bus ID of capture device connected computer.

byPCIDevID

PCI device ID of capture device connected computer.

byLinkType

Data transmission standard negotiated by capture device and PCI-e slots.

byLinkWidth

The lane number of capture device connected to PCI-e slots.

byBoardIndex

The value of board index on capture device.

wMaxPayloadSize

The maximum payload size on capture device.

wMaxReadRequestSize

The maximum payload size on capture device.

cbTotalMemorySize

The total memory size on capture device.

cbFreeMemorySize

The free memory size on capture device.

2.2.3 MWCAP_VIDEO_CAPS structure

The structure describes capacity of video capturing.

Syntax

```
typedef struct _MWCAP_VIDEO_CAPS {  
    DWORD dwCaps;  
    WORD wMaxInputWidth;  
    WORD wMaxInputHeight;  
    WORD wMaxOutputWidth;  
    WORD wMaxOutputHeight;  
} MWCAP_VIDEO_CAPS;
```

Members

dwCaps

Capacity of video capturing.

wMaxInputWidth

The maximum width of input video.

wMaxInputHeight

The maximum height of input video.

wMaxOutputWidth

The maximum width of output video.

wMaxOutputHeight

The maximum height of output video.

2.2.4 MWCAP_AUDIO_CAPS structure

The structure describes capacity of audio capturing.

Syntax

```
typedef struct _MWCAP_AUDIO_CAPS {  
    DWORD dwCaps;  
} MWCAP_AUDIO_CAPS;
```

Members

dwCaps

Capacity of audiocapturing.

2.2.5 MWCAP_FIRMWARE_STORAGE structure

The structure describes firmware storage information.

Syntax

```
typedef struct _MWCAP_FIRMWARE_STORAGE {
    DWORD cbStorage;
    DWORD cbEraseBlock;
    DWORD cbProgramBlock;
    DWORD cbHeaderOffset;
} MWCAP_FIRMWARE_STORAGE;
```

Members

cbStorage

Size of firmware storage.

cbEraseBlock

Size of erase block.

cbProgramBlock

Size of program block.

cbHeaderOffset

Offset of firmware header.

2.2.6 MWCAP_FIRMWARE_ERASE structure

The structure describes the storage area to be erased.

Syntax

```
typedef struct _MWCAP_FIRMWARE_ERASE {
    DWORD cbOffset;
    DWORD cbErase;
} MWCAP_FIRMWARE_ERASE;
```

Members

cbOffset

Offset of the storage area to be erased.

cbErase

Size of storage area to be erased.

2.2.7 MWCAP_SDI_SPECIFIC_STATUS structure

The structure describes SDI signal status.

Syntax

```
typedef struct _MWCAP_SDI_SPECIFIC_STATUS {
    SDI_TYPE sdiType;
    SDI_SCANNING_FORMAT sdiScanningFormat;
    SDI_BIT_DEPTH sdiBitDepth;
    SDI_SAMPLING_STRUCT sdiSamplingStruct;
    BOOLEAN bST352DataValid;
```

DWORD	dwST352Data;
} MWCAP_SDI_SPECIFIC_STATUS;	

Members

sdiType

SDI signal type.

sdiScanningFormat

SDI scanning format

sdiBitDepth

SDI bit depth

sdiSamplingStruct

SDI sampling data struct

bST352DataValid

Whether ST352 data is valid

dwST352Data

ST352 data

2.2.8 MWCAP_HDMI_VIDEO_TIMING structure

The structure describes HDMI video signal timing

Syntax

typedef struct _MWCAP_HDMI_VIDEO_TIMING {	
BOOLEAN	bInterlaced;
DWORD	dwFrameDuration;
WORD	wHSyncWidth;
WORD	wHFrontPorch;
WORD	wHBackPorch;
WORD	wHActive;
WORD	wHTotalWidth;
WORD	wField0VSyncWidth;
WORD	wField0VFrontPorch;
WORD	wField0VBackPorch;
WORD	wField0VActive;
WORD	wField0VTotalHeight;
WORD	wField1VSyncWidth;
WORD	wField1VFrontPorch;
WORD	wField1VBackPorch;
WORD	wField1VActive;
WORD	wField1VTotalHeight;
} MWCAP_HDMI_VIDEO_TIMING;	

Members

bInterlaced

Whether is interlaced.

dwFrameDuration

Frame duration

wHSyncWidth

Horizontal sync width

wHFrontPorch

Width of horizontal front porch

wHBackPorch

Width of horizontal back porch

wHActive

Horizontal valid width

wHTotalWidth

Horizontal total width

wField0VSyncWidth

Vertical sync width of the field 0

wField0VFrontPorch

Vertical front porch width of the field 0

wField0VBackPorch

Vertical back porch width of the field 0

wField0VActive

Vertical valid width of the field 0

wField0VTotalHeight

Vertical total height of the field 0

wField1VSyncWidth

Vertical sync width of the field 1

wField1VFrontPorch

Vertical front porch width of the field 1

wField1VBackPorch

Vertical back porch width of the field 1

wField1VActive

Vertical valid width of the field 1

wField1VTotalHeight

Vertical total height of the field 1

2.2.9 MWCAP_HDMI_SPECIFIC_STATUS structure

The structure describes HDMI signal status

Syntax

```
typedef struct _MWCAP_HDMI_SPECIFIC_STATUS {  
    BOOLEAN                                bHDMIMode;  
    BOOLEAN                                bHDCP;  
    BYTE                                   byBitDepth;  
    HDMI_PXIEL_ENCODING                    pixelEncoding;  
}
```

BYTE	byVIC;
BOOLEAN	bITContent;
BOOLEAN	b3DFormat;
BYTE	by3DStructure;
BYTE	bySideBySideHalfSubSampling;
MWCAP_HDMI_VIDEO_TIMING	videoTiming;
} MWCAP_HDMI_SPECIFIC_STATUS;	

Members

bHDMIMode

Whether is HDMI mode.

bHDCP

Whether is HDCP encrypted signal

byBitDepth

Bit depth

pixelEncoding

Pixel format

byVIC

Video identify code in EDID to specify standard resolution and timing

bITContent

IT Content flag

b3DFormat

Whether is 3D format

by3DStructure

3D structure

bySideBySideHalfSubSampling

Side-by-side half sub sampling

videoTiming

Video timing

2.2.10 MWCAP_COMPONENT_SPECIFIC_STATUS structure

The structure describes Component signal status

Syntax

typedef struct _MWCAP_COMPONENT_SPECIFIC_STATUS {	
MWCAP_VIDEO_SYNC_INFO	syncInfo;
BOOLEAN	bTriLevelSync;
MWCAP_VIDEO_TIMING	videoTiming;
MWCAP_VIDEO_TIMING_SETTINGS	videoTimingSettings;
} MWCAP_COMPONENT_SPECIFIC_STATUS;	

Members

syncInfo

Video sync information.

bTriLevelSync

Whether is trilevel sync.

videoTiming

Video timing information.

videoTimingSettings

Video timing settings.

2.2.11 MWCAP_CVBS_YC_SPECIFIC_STATUS structure

The structure describes CVBS_YC signal status

Syntax

```
typedef struct _MWCAP_CVBS_YC_SPECIFIC_STATUS {  
    MWCAP_SD_VIDEO_STANDARD                standard;  
    BOOLEAN                                b50Hz;  
} MWCAP_CVBS_YC_SPECIFIC_STATUS;
```

Members

Standard

Defines video standard used.

b50Hz

Whether scanning frequency is 50 Hz.

2.2.12 MWCAP_INPUT_SPECIFIC_STATUS structure

The structure describes input signal status

Syntax

```
typedef struct _MWCAP_INPUT_SPECIFIC_STATUS {  
    BOOLEAN                                bValid;  
    DWORD                                dwVideoInputType;  
    union {  
        MWCAP_SDI_SPECIFIC_STATUS        sdiStatus;  
        MWCAP_HDMI_SPECIFIC_STATUS        hdmiStatus;  
        MWCAP_COMPONENT_SPECIFIC_STATUS    vgaComponentStatus;  
        MWCAP_CVBS_YC_SPECIFIC_STATUS      cvbsYcStatus;  
    };  
} MWCAP_INPUT_SPECIFIC_STATUS;
```

Members

bValid

Defines whether such input signal status is valid.

dwVideoInputType

Signal type of the input video .

sdiStatus

SDI signal status.

hdmiStatus

HDMI signal status.

vgaComponentStatus

VGA component signal status.

cvbsYcStatus

CVBS-YC signal status.

2.2.13 MWCAP_VIDEO_SIGNAL_STATUS structure

The structure describes video signal status

Syntax

```
typedef struct _MWCAP_VIDEO_SIGNAL_STATUS {
    MWCAP_VIDEO_SIGNAL_STATE          state;
    int                                x;
    int                                y;
    int                                cx;
    int                                cy;
    int                                cxTotal;
    int                                cyTotal;
    BOOLEAN                            bInterlaced;
    DWORD                              dwFrameDuration;
    int                                nAspectX;
    int                                nAspectY;
    BOOLEAN                            bSegmentedFrame;
    MWCAP_VIDEO_FRAME_TYPE             frameType;
    MWCAP_VIDEO_COLOR_FORMAT           colorFormat;
    MWCAP_VIDEO_QUANTIZATION_RANGE     quantRange;
    MWCAP_VIDEO_SATURATION_RANGE       satRange;
} MWCAP_VIDEO_SIGNAL_STATUS;
```

Members**state**

Defines the accessibility of the video signal.

x

Starting position in the horizontal direction.

y

Starting position in the vertical direction.

cx

Video frame width.

cy
Video frame height.

cxTotal
Total width.

cyTotal
Total height.

bInterlaced
Whether is interlaced signal.

dwFrameDuration
Video frame duration.

nAspectX
Width value in aspect ratio.

nAspectY
Height value in aspect ratio.

bSegmentedFrame
Whether is segmented frame.

frameType
Video frame type.

colorFormat
Video color format.

quantRange
Quantization range.

satRange
Saturation range.

2.2.14 MWCAP_AUDIO_SIGNAL_STATUS structure

The structure describes audio signal status

Syntax

```
typedef struct _MWCAP_AUDIO_SIGNAL_STATUS {
    WORD                wChannelValid;
    BOOLEAN              bLPCM;
    BYTE                cBitsPerSample;
    DWORD               dwSampleRate;
    BOOLEAN              bChannelStatusValid;
    IEC60958_CHANNEL_STATUS channelStatus;
} MWCAP_AUDIO_SIGNAL_STATUS;
```

Members

wChannelValid

Defines valid channels mask. The lowest bit defines whether channel 1 and 2 are valid. The

second bit defines whether channel 3 and 4 are valid. The third bit defines whether channel 5 and 6 are valid. The fourth bit defines whether channel 7 and 8 are valid.

bLPCM

Indicates whether is LPCM format.

cBitsPerSample

Bit depth of each audio sample.

dwSampleRate

Sample rate.

bChannelStatusValid

Indicates whether the following channel status is valid.

channelStatus

Audio channel status.

2.2.15 MWCAP_TIMER_EXPIRE_TIME structure

The structure describes audio signal status

Syntax

```
typedef struct _MWCAP_TIMER_EXPIRE_TIME {  
    MWCAP_PTR64                pvTimer;  
    LONGLONG                   llExpireTime;  
} MWCAP_TIMER_EXPIRE_TIME;
```

Members

pvTimer

Handle of timer.

llExpireTime

The expired time.

2.2.16 MWCAP_KSPROPERTY_NOTIFY_REGISTRATION_S

The structure is used to register notification

Syntax

```
typedef struct _MWCAP_KSPROPERTY_NOTIFY_REGISTRATION_S {  
    KSPROPERTY                 Property;  
    MWCAP_PTR64                hEvent;  
    ULONGLONG                  ullEnableBits;  
} MWCAP_KSPROPERTY_NOTIFY_REGISTRATION_S;
```

Members

Property

Retrives KSPROPERTY object.

hEvent

Event handle to be registered.

ullEnableBits

Event type mask. Reference to MWCAP_NOTIFY_INPUT_SOURCE_START_SCAN and so on.

2.2.17 MWCAP_SMPTE_TIMECODE structure

The structure defines SMPTE time code

Syntax

```
typedef struct _MWCAP_SMPTE_TIMECODE {  
    BYTE                byFrames;  
    BYTE                bySeconds;  
    BYTE                byMinutes;  
    BYTE                byHours;  
} MWCAP_SMPTE_TIMECODE;
```

Members

byFrames

Frame number.

bySeconds

Seconds.

byMinutes

Minutes.

byHours

Hours.

2.2.18 MWCAP_VIDEO_BUFFER_INFO structure

The structure defines video buffer information

Syntax

```
typedef struct _MWCAP_VIDEO_BUFFER_INFO {  
    DWORD                cMaxFrames;  
    BYTE                iNewestBuffering;  
    BYTE                iBufferingFieldIndex;  
    BYTE                iNewestBuffered;  
    BYTE                iBufferedFieldIndex;  
    BYTE                iNewestBufferedFullFrame;  
    DWORD                cBufferedFullFrames;  
} MWCAP_VIDEO_BUFFER_INFO;
```

Members

cMaxFrames

Maximum frame count in buffer on board.

iNewestBuffering

Index of the newest buffering slice. A video frame may contain multi-slices.

iBufferingFieldIndex

Index of the newest buffering field.

iNewestBuffered

Index of the newest buffered slice .

iBufferedFieldIndex

Index of the newest buffering field.

iNewestBufferedFullFrame

Index of the newest buffered frame.

cBufferedFullFrames

count of buffered frames.

2.2.19 MWCAP_VIDEO_FRAME_INFO structure

The structure defines video frame information

Syntax

```
typedef struct _MWCAP_VIDEO_FRAME_INFO {
    MWCAP_VIDEO_FRAME_STATE          state;
    BOOLEAN                          bInterlaced;
    BOOLEAN                          bSegmentedFrame;
    BOOLEAN                          bTopFieldFirst;
    BOOLEAN                          bTopFieldInverted;
    int                              cx;
    int                              cy;
    int                              nAspectX;
    int                              nAspectY;
    LONGLONG                         allFieldStartTimes[2];
    LONGLONG                         allFieldBufferedTimes[2];
    MWCAP_SMPTE_TIMECODE             aSMPTETimeCodes[2];
} MWCAP_VIDEO_FRAME_INFO;
```

Members

state

Indicate the video frame state.

bInterlaced

Whether is interlaced.

bSegmentedFrame

Whether is segmented frame.

bTopFieldFirst

Whether top field comes first .

bTopFieldInverted

Whether top field is inverted..

cx

Video frame width.

cy

Video frame height.

nAspectX

Width in aspect ratio.

nAspectY

Height in aspect ratio.

allFieldStartTimes

Start times of the top field and bottom field.

allFieldBufferedTimes

Buffered times of the top field and bottom field.

aSMPTETimeCodes

Time codes of the top field and bottom field.

2.2.20 MWCAP_VIDEO_CAPTURE_OPEN structure

The structure defines event handle to start capture

Syntax

```
typedef struct _MWCAP_VIDEO_CAPTURE_OPEN {
    MWCAP_PTR64                                     hEvent;
} MWCAP_VIDEO_CAPTURE_OPEN;
```

Members**hEvent**

Event handle.

2.2.21 MWCAP_VIDEO_CAPTURE_FRAME structure

The structure defines capture parameters of video frame

Syntax

```
typedef struct _MWCAP_VIDEO_CAPTURE_FRAME {
    DWORD                                     dwFOURCC;
    WORD                                      cx;
    WORD                                      cy;
    int                                       nAspectX;
    int                                       nAspectY;
    MWCAP_VIDEO_COLOR_FORMAT                 colorFormat;
    MWCAP_VIDEO_QUANTIZATION_RANGE           quantRange;
}
```

MWCAP_VIDEO_SATURATION_RANGE	satRange;
SHORT	sContrast;
SHORT	sBrightness;
SHORT	sSaturation;
SHORT	sHue;
RECT	rectSource;
RECT	rectTarget;
MWCAP_VIDEO_DEINTERLACE_MODE	deinterlaceMode;
MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODE	aspectRatioConvertMode;
char	iSrcFrame;
MWCAP_PTR64	pOSDImage;
RECT	aOSDRects[MWCAP_VIDEO_MAX_NUM_OSD_RECTS];
int	cOSDRects;
BOOLEAN	bPhysicalAddress;
union {	
MWCAP_PTR64	pvFrame;
LARGE_INTEGER	liPhysicalAddress;
};	
DWORD	cbFrame;
DWORD	cbStride;
BOOLEAN	bBottomUp;
WORD	cyPartialNotify;
DWORD	dwProcessSwitchs;
MWCAP_PTR64	pvContext;
} MWCAP_VIDEO_CAPTURE_FRAME;	

Members

dwFOURCC

Colors space. Reference to MWFOURCC.h.

cx

Capture width.

cy

Capture height.

nAspectX

Width in aspect ratio.

nAspectY

Height in aspect ratio.

colorFormat

Colors format standard.

quantRange

Quantization range.

satRange

Saturation range.

sContrast

Contrast.

sBrightness
Brightness.

sSaturation
Saturation.

sHue
Hue.

rectSource
Source rect to capture from.

rectTarget
Target rect to capture to.

deinterlaceMode
De-interlaced mode.

aspectRatioConvertMode
Aspect ratio convert mode.

iSrcFrame
Index of source frame to be capture.

pOSDImage
OSD image handle from invoking interface MWCAP_KSPROPERTY_VIDEO_CREATE_IMAGE.

aOSDRects
Target rect to render OSD image.

cOSDRects
Count of target rect to render OSD image.

bPhysicalAddress
Whether capture video frame data to physical address.

pvFrame
Memory address to store captured video frame data.

liPhysicalAddress
Physical address to store captured video frame data.

cbFrame
Frame size.

cbStride
Stride.

bBottomUp
Whether is bottom-up.

cyPartialNotify
Count of partial notifies.

dwProcessSwitchs
Mask of video processes. Reference to MWCAP_VIDEO_PROCESS_xx.

pvContext
Context of capturing such video frame.

2.2.22 MWCAP_VIDEO_CAPTURE_STATUS structure

The structure defines capture status

Syntax

```
typedef struct _MWCAP_VIDEO_CAPTURE_STATUS {  
    MWCAP_PTR64                pvContext;  
    BOOLEAN                    bPhysicalAddress;  
    union {  
        MWCAP_PTR64            pvFrame;  
        LARGE_INTEGER           liPhysicalAddress;  
    };  
    int                         iFrame;  
    BOOLEAN                    bFrameCompleted;  
    WORD                       cyCompleted;  
    WORD                       cyCompletedPrev;  
} MWCAP_VIDEO_CAPTURE_STATUS;
```

Members

pvContext

Context of video capturing.

bPhysicalAddress

Whether use physical address to store captured data.

pvFrame

Memory address to store captured data.

liPhysicalAddress

Physical address to store captured data.

iFrame

the index of the frame capturing

bFrameCompleted

Whether a whole frame is captured.

cyCompleted

Count of frames captured.

cyCompletedPrev

Count of frames captured previous.

2.2.23 MWCAP_AUDIO_CAPTURE_FRAME structure

The structure defines audio capture frame info

Syntax

```
typedef struct _MWCAP_AUDIO_CAPTURE_FRAME {  
    DWORD                    cFrameCount;  
    DWORD                    iFrame;  
    DWORD                    dwSyncCode;
```

DWORD	dwReserved;
LONG LONG	llTimestamp;
DWORD	adwSamples[MWCAP_AUDIO_SAMPLES_PER_FRAME* MWCAP_AUDIO_MAX_NUM_CHANNELS];
} MWCAP_AUDIO_CAPTURE_FRAME;	

Members

cFrameCount

Count of frames buffered.

iFrame

Index of such frame.

dwSyncCode

Sync code of audio frame data.

dwReserved

Reserved DWORD value.

llTimestamp

Time stamp of such audio frame.

adwSamples

Audio sampling data. Each sample is 32 bits wide. cBitsPerSample of high bits are valid. Sample layout: 0L, 1L, 2L, 3L, 0R, 1R, 2R, 3R.

2.2.24 MWCAP_VIDEO_ASPECT_RATIO structure

The structure defines video aspect ratio

Syntax

typedef struct _MWCAP_VIDEO_ASPECT_RATIO {	
int	nAspectX;
int	nAspectY;
} MWCAP_VIDEO_ASPECT_RATIO;	

Members

nAspectX

Aspect width.

nAspectY

Aspect height.

2.2.25 MWCAP_VIDEO_CONNECTION_FORMAT structure

The structure defines video connection format.

Syntax

typedef struct _MWCAP_VIDEO_CONNECTION_FORMAT {	
BOOLEAN	bConnected;

LONG	cx;
LONG	cy;
DWORD	dwFrameDuration;
DWORD	dwFOURCC;
int	nAspectX;
int	nAspectY;
MWCAP_VIDEO_COLOR_FORMAT	colorFormat;
MWCAP_VIDEO_QUANTIZATION_RANGE	quantRange;
MWCAP_VIDEO_SATURATION_RANGE	satRange;
} MWCAP_VIDEO_CONNECTION_FORMAT;	

Members

bConnected

Indicates whether is connected.

cx

Video frame width.

cy

Video frame height.

dwFrameDuration

Video frame duration.

dwFOURCC

Color space. Reference to MWFOURCC.h.

nAspectX

Width value in aspect ratio.

nAspectY

Height value in aspect ratio.

colorFormat

Video color format standard.

quantRange

Quantization range.

satRange

Saturation range.

2.2.26 MWCAP_VIDEO_PROCESS_SETTINGS structure

The structure defines video process settings.

Syntax

typedef struct _MWCAP_VIDEO_PROCESS_SETTINGS {	
DWORD	dwProcessSwitchs;
RECT	rectSource;
int	nAspectX;
int	nAspectY;

BOOLEAN	bLowLatency;
MWCAP_VIDEO_COLOR_FORMAT	colorFormat;
MWCAP_VIDEO_QUANTIZATION_RANGE	quantRange;
MWCAP_VIDEO_SATURATION_RANGE	satRange;
MWCAP_VIDEO_DEINTERLACE_MODE	deinterlaceMode;
MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODE	aspectRatioConvertMode;
} MWCAP_VIDEO_PROCESS_SETTINGS;	

Members

dwProcessSwitchs

Mask of video processes. Reference to MWCAP_VIDEO_PROCESS_xx.

rectSource

Source area to process.

nAspectX

Width value in aspect ratio.

nAspectY

Height value in aspect ratio.

bLowLatency

Whether use lowlatency mode.

colorFormat

Video color format standard.

quantRange

Quantization range.

satRange

Saturation range.

deinterlaceMode

De-interlaced mode.

aspectRatioConvertMode

Aspect ratio convert mode.

2.2.27 MWCAP_KSPROPERTY_VIDEO_CREATE_IMAGE_S structure

The structure defines parameters to create image on board.

Syntax

typedef struct _MWCAP_KSPROPERTY_VIDEO_CREATE_IMAGE_S {	
KSPROPERTY	Property;
WORD	cx;
WORD	cy;
} MWCAP_KSPROPERTY_VIDEO_CREATE_IMAGE_S;	

Members

Property

KSPROPERTY object.

cx

Image width.

cy

Image height.

2.2.28 MWCAP_VIDEO_UPLOAD_IMAGE structure

The structure defines parameters to upload image on board.

Syntax

```
typedef struct _MWCAP_VIDEO_UPLOAD_IMAGE {  
    MWCAP_PTR64                pvDestImage;  
    MWCAP_VIDEO_COLOR_FORMAT    cfDest;  
    WORD                        xDest;  
    WORD                        yDest;  
    WORD                        cxDest;  
    WORD                        cyDest;  
    MWCAP_VIDEO_QUANTIZATION_RANGE quantRangeDest;  
    MWCAP_VIDEO_SATURATION_RANGE satRangeDest;  
    BOOLEAN                     bSrcPhysicalAddress;  
    union {  
        MWCAP_PTR64                pvSrcFrame;  
        LARGE_INTEGER               liSrcPhysicalAddress;  
    };  
    DWORD                        cbSrcFrame;  
    DWORD                        cbSrcStride;  
    WORD                         cxSrc;  
    WORD                         cySrc;  
    BOOLEAN                      bSrcBottomUp;  
    BOOLEAN                      bSrcPixelAlpha;  
    BOOLEAN                      bSrcPixelXBGR;  
} MWCAP_VIDEO_UPLOAD_IMAGE;
```

Members

pvDestImage

Handle of target image.

cfDest

Color format standard.

xDest

Horizontal starting position of target area.

yDest

Vertical starting position of target area.

cxDest

Width of target area.

cyDest

Height of target area.

quantRangeDest

Quantization range of destination image.

satRangeDest

Saturation range of destination image.

bSrcPhysicalAddress

Whether source video uses physical address.

pvSrcFrame

Memory address of source video data.

liSrcPhysicalAddress

Physical address of source video data.

cbSrcFrame

Total length of source video data.

cbSrcStride

Stride of source video data.

cxSrc

Width of source video.

cySrc

Height of source video.

bSrcBottomUp

Whether source video is bottom-up.

bSrcPixelAlpha

Whether source video pixel has alpha component.

bSrcPixelXBGR

Whether source video pixel format is XBGR..

2.2.29 MWCAP_VIDEO_OSD_SETTINGS structure

The structure defines OSD settings.

Syntax

```
typedef struct _MWCAP_VIDEO_OSD_SETTINGS {
    BOOLEAN                                     bEnable;
    WCHAR                                     wszPNGFilePath[_MAX_PATH];
} MWCAP_VIDEO_OSD_SETTINGS;
```

Members**bEnable**

Whether to use OSD.

wszPNGFilePath

File path of png image to upload.

2.2.30 MWCAP_VIDEO_OSD_IMAGE structure

The structure defines OSD image information.

Syntax

```
typedef struct _MWCAP_VIDEO_OSD_IMAGE {  
    MWCAP_PTR64          pvOSDImage;  
    RECT                 aOSDRects[MWCAP_VIDEO_MAX_NUM_OSD_RECTS];  
    int                  cOSDRects;  
} MWCAP_VIDEO_OSD_IMAGE;
```

Members

pvOSDImage

OSD image handle.

aOSDRects

Target areas to blend such image.

cOSDRects

Count of target areas to blend such image.

2.2.31 MWCAP_VIDEO_CUSTOM_TIMING structure

The structure defines custom video timing.

Syntax

```
typedef struct _MWCAP_VIDEO_CUSTOM_TIMING {  
    MWCAP_VIDEO_SYNC_INFO          syncInfo;  
    MWCAP_VIDEO_TIMING_SETTINGS    videoTimingSettings;  
} MWCAP_VIDEO_CUSTOM_TIMING;
```

Members

syncInfo

Video sync info.

videoTimingSettings

Settings of video timing.

2.2.32 MWCAP_VIDEO_SYNC_INFO structure

The structure describes sync information of video timing

Syntax

```
typedef struct _MWCAP_VIDEO_SYNC_INFO {
    BYTE                bySyncType;
    BOOLEAN              bHSPolarity;
    BOOLEAN              bVSPolarity;
    BOOLEAN              bInterlaced;
    DWORD                dwFrameDuration;
    WORD                 wVSyncLineCount;
    WORD                 wFrameLineCount;
} MWCAP_VIDEO_SYNC_INFO;
```

Members

bySyncType

Sync type of video timing.

bHSPolarity

Horizontal sync polarity of video timing.

bVSPolarity

Vertical sync polarity of video timing.

bInterlaced

If video timing is interlaced.

dwFrameDuration

Frame duration of video timing.

wVSyncLineCount

Count of vertical sync lines in video timing.

wFrameLineCount

Count of frame lines in video timing.

2.2.33 MWCAP_VIDEO_TIMING structure

The structure describes video channel timing information.

Syntax

```
typedef struct _MWCAP_VIDEO_TIMING {
    DWORD                dwType;
    DWORD                dwPixelClock;
    BOOLEAN              bInterlaced;
    BYTE                bySyncType;
    BOOLEAN              bHSPolarity;
    BOOLEAN              bVSPolarity;
    WORD                 wHActive;
    WORD                 wHFrontPorch;
    WORD                 wHSyncWidth;
    WORD                 wHBackPorch;
    WORD                 wVActive;
    WORD                 wVFrontPorch;
}
```


WORD	wVSyncWidth;
WORD	wVBackPorch;
} MWCAP_VIDEO_TIMING;	

Members

dwType

Type of video channel timing.

dwPixelClock

Pixel clock of video channel timing.

bInterlaced

If video channel timing is interlaced.

bySyncType

Sync type of video channel timing.

bHSPolarity

If horizon sync polarity is positive.

bVSPolarity

If horizontal sync polarity is positive.

wHActive

Horizontal active video time in video channel timing.

wHFrontPorch

Horizontal front porch in video channel timing.

wHSyncWidth

Horizontal sync width in video channel timing.

wHBackPorch

Horizontal back porch in video channel timing.

wVActive

Vertical active video time in video channel timing.

wVFrontPorch

Vertical front porch in video channel timing.

wVSyncWidth

Vertical sync width in video channel timing.

wVBackProch

Vertical back porch in video channel timing.

2.2.34 MWCAP_VIDEO_TIMING_SETTINGS structure

The structure describes video timing settings.

Syntax

typedef struct _MWCAP_VIDEO_TIMING_SETTINGS {	
WORD	wAspectX;
WORD	wAspectY;
WORD	x;
WORD	y;

WORD	cx;
WORD	cy;
WORD	cxTotal;
BYTE	byClampPos;
} MWCAP_VIDEO_TIMING_SETTINGS;	

Members

wAspectX

Width value in aspect ratio

wAspectY

Height value in aspect ratio

x

Startting position in horizontal direction

y

Startting position in vertical direction

cx

Width.

cy

Height.

cxTotal

Totall width inhorizontal direction

byClampPos

Clamping position.

2.2.35 MWCAP_KSPROPERTY_DWORD_S structure

The structure appends a parameter after the KSPROPERTY object

Syntax

typedef struct _MWCAP_KSPROPERTY_DWORD_S {	
KSPROPERTY	Property;
DWORD	dwParameter;
} MWCAP_KSPROPERTY_DWORD_S;	

Members

Property

KSPROPERTY object to invoke interface

dwParameter

Appended DWORD parameter

2.2.36 MWCAP_KSPROPERTY_PTR64_S structure

The structure appends a parameter after the KSPROPERTY object

Syntax

```
typedef struct _MWCAP_KSPROPERTY_PTR64_S {
    KSPROPERTY                                Property;
    MWCAP_PTR64                               pvParameter;
} MWCAP_KSPROPERTY_PTR64_S;
```

Members

Property

KSPROPERTY object to invoke interface

pvParameter

Appended MWCAP_PTR64 parameter

2.2.37 MWCAP_DWORD_PARAMETER_RANGE structure

The structure defines a range, step and the default value

Syntax

```
typedef struct _MWCAP_DWORD_PARAMETER_RANGE {
    DWORD                                dwMin;
    DWORD                                dwMax;
    DWORD                                dwStep;
    DWORD                                dwDefault;
} MWCAP_DWORD_PARAMETER_RANGE;
```

Members

dwMin

The minimum value

dwMax

The maximum value

dwStep

The step length

dwDefault

The default value

2.2.38 MWCAP_DWORD_PARAMETER_VALUE structure

The structure defines a DWORD parameter value

Syntax

```
typedef struct _MWCAP_DWORD_PARAMETER_VALUE {
    DWORD                                dwFlags;
    DWORD                                dwValue;
} MWCAP_DWORD_PARAMETER_VALUE;
```

Members

dwFlags

Flag

dwValue

Value

3 Functions

3.1 DirectShow Expansion Interface Functions

3.1.1 GetChannelInfo

The method can be used to get capture channel information.

Syntax

```
BOOL GetChannelInfo(  
    MWCAP_CHANNEL_INFO * pChannelInfo  
);
```

Params

pChannelInfo [out]

The capture channel information.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_CHANNEL_INFO structure.

3.1.2 GetFamilyInfo

The method can be used to get device family information.

Syntax

```
BOOL GetFamilyInfo(  
    void *          pvFamilyInfo,  
    ULONG *         pcbFamilyInfo  
);
```

Params

pvFamilyInfo [out]

The capture channel family information. Input MWCAP_PRO_CAPTURE_INFO structure point for Pro Capture Serial Cards

pcbFamilyInfo [in]

The length of pvFamilyInfo memory.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.3 GetVideoCaps

The method can be used to get video channel capacity.

Syntax

BOOL GetVideoCaps(MWCAP_VIDEO_CAPS * pVideoCaps);
--

Params

pVideoCaps [out]

The capability of video channel.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_CAPSstructure.

3.1.4 GetAudioCaps

The method can be used to get audio channel capacity.

Syntax

BOOL GetAudioCaps (MWCAP_AUDIO_CAPS * pAudioCaps);

Params

pAudioCaps [out]

The capability of audio channel.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_AUDIO_CAPS structure.

3.1.5 GetDeviceInstanceID

The method can be used to get channel' s system device ID.

Syntax

```
BOOL GetDeviceInstanceID (  
    WCHAR *      pszInstanceID,  
    ULONG *      pcbInstanceID  
);
```

Params

pszInstanceID [out]

The channel' s system device ID.

pcbInstanceID [in]

The length of pszInstanceID memory.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.6 RegisterNotify

The method can be used to register asynchronous event.

Syntax

```
BOOL RegisterNotify(  
    HANDLE      hEvent,
```

ULONGLONG	ullEnableBits,
MWCAP_PTR64&	pvNotifyEvent
);	

Params

hEvent [in]

The event handle.

ullEnableBits [in]

The value of notify event.

pvNotifyEvent [out]

The asynchronous notifying event.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.7 UnregisterNotify

The method can be used to unregister asynchronous event.

Syntax

BOOL UnregisterNotify(
MWCAP_PTR64	pvNotifyEvent
);	

Params

pvNotifyEvent [in]

The asynchronous notifying event.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.8 GetNotifyStatus

The method can be used to get event notify status.

Syntax

```
BOOL GetNotifyStatus(  
    MWCAP_PTR64    pvNotifyEvent,  
    ULONGLONG&     ullStatusBits  
);
```

Params

pvNotifyEvent [in]

The value of rotary switch on the board.

ullStatusBits [out]

The value of notify event.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.9 GetVideoInputSourceCount

The method can be used to get video channel all input sources count.

Syntax

```
BOOL GetVideoInputSourceCount(  
    DWORD *    pcInputSources  
);
```

Params

pcInputSources [out]

All video channel input sources count.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.10 GetVideoInputSourceArray

The method can be used to get video channel all input sources.

Syntax

```
BOOL GetVideoInputSourceArray(  
    DWORD *    pdwInputSources,  
    DWORD *    pcBufferItems  
);
```

Params

pdwInputSources [out]

The value of video input source.

pcBufferItems [out]

The number of video input source type.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.11 GetAudioInputSourceCount

The method can be used to get audio channel all input sources count.

Syntax

```
BOOL GetAudioInputSourceCount(  
    DWORD *    pcInputSources  
);
```

Params

pcInputSources [in]

All audio channel input sources count.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_CHANNEL_INFO structure.

3.1.12 GetAudioInputSourceArray

The method can be used to get audio channel all input sources.

Syntax

```
BOOL GetAudioInputSourceArray(  
    DWORD *    pdwInputSources,  
    DWORD *    pcBufferItems  
);
```

Params

pdwInputSources [out]

The value of audio input source.

pcBufferItems [out]

The number of audio input souce type.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.13 SetInputSourceScan

The method can be used to set capture channel input source scan status.

Syntax

```
BOOL SetInputSourceScan(  
    BOOLEAN    bInputSourceScan  
);
```

Params

bInputSourceScan [in]

Set whether to automatically scan video signal enabled

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.14 GetInputSourceScan

The method can be used to get capture channel input source scan status.

Syntax

```
BOOL GetInputSourceScan(  
    BOOLEAN *    pbInputSourceScan  
);
```

Params

pbInputSourceScan [out]

Get whether to automatically scan video signal enabled

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.15 GetInputSourceScanState

The method can be used to get capture channel input source scan status.

Syntax

```
BOOL GetInputSourceScanState(  
    BOOLEAN *    pbInputSourceScanning  
);
```

Params

pbInputSourceScanning [out]

The input source scan status

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.16 SetAVInputSourceLink

The method can be used to set current audio channel and video channel link status.

Syntax

```
BOOL SetAVInputSourceLink(  
    BOOLEAN    bAVLink  
);
```

Params

bAVLink [in]

The current audio and video channels are consistent.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.17 GetAVInputSourceLink

The method can be used to get current audio channel and video channel link status.

Syntax

```
BOOL GetAVInputSourceLink(  
    BOOLEAN *    pbAVLink  
);
```

Params

pbAVLink [out]

The current audio and video channels are consistent.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.18 SetVideoInputSource

The method can be used to set video channel current input source.

Syntax

```
BOOL SetVideoInputSource(  
    DWORD    dwInputSource  
);
```

Params

dwInputSource [in]
The video input source type.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.19 GetVideoInputSource

The method can be used to get video channel current input source.

Syntax

```
BOOL GetVideoInputSource(  
    DWORD *    pdwInputSource  
);
```

Params

pdwInputSource [out]
The video input source type.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.20 SetAudioInputSource

The method can be used to set audio channel current input source.

Syntax

```
BOOL SetAudioInputSource(  
    DWORD    dwInputSource  
);
```

Params

dwInputSource [in]
The audio input source type.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.21 GetAudioInputSource

The method can be used to get audio channel current input source.

Syntax

```
BOOL GetAudioInputSource(  
    DWORD *    pdwInputSource  
);
```

Params

pdwInputSource [out]
The audio input source type.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.22 SetEDID

The method can be used to set EDID data to HDMI interface.

Syntax

```
BOOL SetEDID(  
    LPBYTE    pbyEDID,  
    ULONG     cbEDID  
);
```

Params

pbyEDID [in]

EDID data.

cbEDID [in]

The pbyEDID data memory size in bytes.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.23 GetEDID

The method can be used to get EDID data from HDMI interface.

Syntax

```
BOOL GetEDID(  
    LPBYTE    pbyEDID,  
    ULONG *   pcbEDID  
);
```

Params

pbyEDID [out]

EDID data.

cbEDID [out]

The pbyEDID data memory size in bytes.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.24 GetInputSpecificStatus

The method can be used to get input signal status.

Syntax

```
BOOL GetInputSpecificStatus(  
    MWCAP_INPUT_SPECIFIC_STATUS *    pStatus  
);
```

Params

pStatus [out]

Returns the specified input signal status.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_INPUT_SPECIFIC_STATUS structure.

3.1.25 GetVideoSignalStatus

The method can be used to get input video signal status.

Syntax

```
BOOL GetVideoSignalStatus(  
    MWCAP_VIDEO_SIGNAL_STATUS *    pStatus  
);
```

Params

pStatus [out]

Returns the video input signal status.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_SIGNAL_STATUS structure.

3.1.26 GetAudioSignalStatus

The method can be used to get input audio signal status.

Syntax

```
BOOL GetAudioSignalStatus(  
    MWCAP_AUDIO_SIGNAL_STATUS *    pStatus  
);
```

Params

pStatus [out]

Returns the audio input signal status.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_AUDIO_SIGNAL_STATUS structure.

3.1.27 GetHDMIInfoFrameValidFlags

The method can be used to get HDMI InfoFrame valid status.

Syntax

```
BOOL GetHDMIInfoFrameValidFlags(  
    DWORD *    pdwValidFlags  
);
```

Params

pdwValidFlags [out]

HDMI InfoFrame valid status.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.28 GetHDMIInfoFramePacket

The method can be used to get HDMI InfoFrame data.

Syntax

```
BOOL GetHDMIInfoFramePacket(  
    MWCAP_HDMI_INFOFRAME_ID id,  
    HDMI_INFOFRAME_PACKET *    pInfoFramePacket  
);
```

Params

id [in]

Reference MWCAP_HDMI_INFOFRAME_ID.

pInfoFramePacket [out]

HDMI InfoFrame packe information.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of HDMI_INFOFRAME_PACKET structure.

3.1.29 SetVideoInputAspectRatio

The method can be used to set video channel input aspect ratio.

Syntax

```
BOOL SetVideoInputAspectRatio(  
    const MWCAP_VIDEO_ASPECT_RATIO *    pAspectRatio  
);
```

Params

pAspectRatio [in]

The video aspect ratio.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_ASPECT_RATIO structure.

3.1.30 GetVideoInputAspectRatio

The method can be used to get video channel input aspect ratio.

Syntax

```
BOOL GetVideoInputAspectRatio(  
    MWCAP_VIDEO_ASPECT_RATIO *    pAspectRatio  
);
```

Params

pAspectRatio [out]

The video aspect ratio.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_ASPECT_RATIO structure.

3.1.31 SetVideoInputColorFormat

The method can be used to set video channel input color format.

Syntax

```
BOOL SetVideoInputColorFormat(  
    MWCAP_VIDEO_COLOR_FORMAT    colorFormat  
);
```

Params

colorFormat [in]

The video color space format

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_COLOR_FORMAT enumerate.

3.1.32 GetVideoInputColorFormat

The method can be used to get video channel input color format.

Syntax

```
BOOL GetVideoInputColorFormat(  
    MWCAP_VIDEO_COLOR_FORMAT *    pColorFormat  
);
```

Params

pColorFormat [out]

The video color space format

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_COLOR_FORMAT enumerate.

3.1.33 SetVideoInputQuantizationRange

The method can be used to set video channel input quantization range.

Syntax

```
BOOL SetVideoInputQuantizationRange(  
    MWCAP_VIDEO_QUANTIZATION_RANGE    quantRange  
);
```

Params

quantRange [in]

The video input quantization range.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_QUANTIZATION_RANGE enumerate.

3.1.34 GetVideoInputQuantizationRange

The method can be used to get video channel input quantization range.

Syntax

BOOL GetVideoInputQuantizationRange(MWCAP_VIDEO_QUANTIZATION_RANGE * pQuantRange);

Params

pQuantRange [out]

Input video signal quantization range.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_QUANTIZATION_RANGE enumerate.

3.1.35 GetVideoCaptureConnectionFormat

The method can be used to get current channel video capture connection format.

Syntax

BOOL GetVideoCaptureConnectionFormat(MWCAP_VIDEO_CONNECTION_FORMAT * pFormat);

Params

pFormat [out]

The video capture format.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_CONNECTION_FORMAT enumerate.

3.1.36 SetVideoCaptureProcessPreset

The method can be used to set current video capture process preset.

Syntax

```
BOOL SetVideoCaptureProcessPreset(  
    const MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [in]

Preset the video capture process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.37 GetVideoCaptureProcessPreset

The method can be used to get current video capture process preset.

Syntax

```
BOOL GetVideoCaptureProcessPreset(  
    MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [out]

Preset the video capture process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.38 SetVideoCaptureProcessSettings

The method can be used to set current video capture process settings.

Syntax

```
BOOL SetVideoCaptureProcessSettings(  
    const MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [in]

The video capture process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.39 GetVideoCaptureProcessSettings

The method can be used to get current video capture process settings.

Syntax

```
BOOL GetVideoCaptureProcessSettings(  
    MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [out]

The video capture process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.40 SetVideoCaptureOSDPreset

The method can be used to set current video capture OSD preset.

Syntax

```
BOOL SetVideoCaptureOSDPreset(  
    const MWCAP_VIDEO_OSD_SETTINGS *    pSettings  
);
```

Params

pSettings [in]

Preset the video capture OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.41 GetVideoCaptureOSDPreset

The method can be used to get current video capture OSD preset.

Syntax

```
BOOL GetVideoCaptureOSDPreset(  
    MWCAP_VIDEO_OSD_SETTINGS *    pSettings  
);
```

Params

pSettings [out]

Preset the video capture OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.42 SetVideoCaptureOSDSettings

The method can be used to set current video capture OSD settings.

Syntax

```
BOOL SetVideoCaptureOSDSettings(  
    const MWCAP_VIDEO_OSD_SETTINGS *  pSettings  
);
```

Params

pSettings [in]

The video capture OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.43 GetVideoCaptureOSDSettings

The method can be used to get current video capture OSD settings.

Syntax

```
BOOL GetVideoCaptureOSDSettings(  
    MWCAP_VIDEO_OSD_SETTINGS *  pSettings  
);
```

Params

pSettings [out]

The video capture OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.44 GetVideoPreviewConnectionFormat

The method can be used to get current video preview connection format.

Syntax

```
BOOL GetVideoPreviewConnectionFormat(  
    MWCAP_VIDEO_CONNECTION_FORMAT *    pFormat  
);
```

Params

pFormat [out]

The video preview format.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_CONNECTION_FORMAT structure.

3.1.45 SetVideoPreviewProcessPreset

The method can be used to set current video preview process preset.

Syntax

```
BOOL SetVideoPreviewProcessPreset(  
    const MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [in]

Preset the video preview process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.46 GetVideoPreviewProcessPreset

The method can be used to get current video preview process preset.

Syntax

```
BOOL GetVideoPreviewProcessPreset(  
    MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [out]

Preset the video preview process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.47 SetVideoPreviewProcessSettings

The method can be used to set current video preview process settings.

Syntax

```
BOOL SetVideoPreviewProcessSettings(  
    const MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [in]

The video preview process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.48 GetVideoPreviewProcessSettings

The method can be used to get current video preview process settings.

Syntax

```
BOOL GetVideoPreviewProcessSettings(  
    MWCAP_VIDEO_PROCESS_SETTINGS *    pSettings  
);
```

Params

pSettings [out]

The video preview process settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_PROCESS_SETTINGS structure.

3.1.49 SetVideoPreviewOSDPreset

The method can be used to set current video preview OSD preset.

Syntax

```
BOOL SetVideoPreviewOSDPreset(  
    const MWCAP_VIDEO_OSD_SETTINGS *    pSettings  
);
```

Params

pSettings [in]

The video preview OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.50 GetVideoPreviewOSDPreset

The method can be used to get current video preview OSD preset.

Syntax

```
BOOL GetVideoPreviewOSDPreset(  
    MWCAP_VIDEO_OSD_SETTINGS *    pSettings  
);
```

Params

pSettings [out]

The video preview OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.51 SetVideoPreviewOSDSettings

The method can be used to set current video preview OSD settings.

Syntax

```
BOOL SetVideoPreviewOSDSettings(  
    const MWCAP_VIDEO_OSD_SETTINGS *    pSettings  
);
```

Params

pSettings [in]

The video preview OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.52 GetVideoPreviewOSDSettings

The method can be used to get current video preview OSD settings.

Syntax

```
BOOL GetVideoPreviewOSDSettings(  
    MWCAP_VIDEO_OSD_SETTINGS *    pSettings  
);
```

Params

pSettings [out]

The video preview OSD settings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_OSD_SETTINGS structure.

3.1.53 SetVideoAutoHAlign

The method can be used to set current video signal horizontal align.

Syntax

```
BOOL SetVideoAutoHAlign(  
    BOOLEAN    bAutoHAlign  
);
```

Params

bAutoHAlign [in]

Whether to set current video signal horizontal align.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.54 GetVideoAutoHAlign

The method can be used to get current video signal horizontal align.

Syntax

```
BOOL GetVideoAutoHAlign(  
    BOOLEAN *    pbAutoHAlign  
);
```

Params

pbAutoHAlign [out]

Whether to set current video signal horizontal align.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.55 SetVideoSamplingPhase

The method can be used to set current video sampling phase.

Syntax

```
BOOL SetVideoSamplingPhase(  
    BYTE    bySamplingPhase  
);
```

Params

bySamplingPhase [in]

The value of video smpling phase.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.56 GetVideoSamplingPhase

The method can be used to get current video sampling phase.

Syntax

```
BOOL GetVideoSamplingPhase(  
    BYTE *    pbySamplingPhase  
);
```

Params

pbySamplingPhase [out]

The value of video smpling phase.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.57 SetVideoSamplingPhaseAutoAdjust

The method can be used to set current video sampling phase auto adjust.

Syntax

```
BOOL SetVideoSamplingPhaseAutoAdjust(  
    BOOLEAN    bAuto  
);
```

Params

bAuto [in]

Whether to adjust automatically the video sampling phase.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.58 GetVideoSamplingPhaseAutoAdjust

The method can be used to get current video sampling phase auto adjust.

Syntax

```
BOOL GetVideoSamplingPhaseAutoAdjust(  
    BOOLEAN *    pbAuto  
);
```

Params

pbAuto [out]

Whether to adjust automatically the video sampling phase.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.59 SetVideoTiming

The method can be used to set current video timing.

Syntax

```
BOOL SetVideoTiming(  
    MWCAP_VIDEO_TIMING *    pTiming  
);
```

Params

pTiming [in]

The video timing.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_TIMINGstructure.

3.1.60 GetPreferredVideoTimings

The method can be used to get current preferred video timings.

Syntax

```
BOOL GetPreferredVideoTimings(  
    MWCAP_VIDEO_TIMING *    pTiming,  
    DWORD *                 pcTimings  
);
```

Params

pTiming [out]

The preferred video timings.

pcTimings [out]

The number of preferred video timings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_TIMING structure.

3.1.61 SetCustomVideoTiming

The method can be used to set current custom video timing.

Syntax

```
BOOL SetCustomVideoTiming(  
    MWCAP_VIDEO_CUSTOM_TIMING *    pTiming  
);
```

Params

pTiming [in]

The current custom video timing.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_CUSTOM_TIMING structure.

3.1.62 GetCustomVideoTimingsCount

The method can be used to get custom video all timings count.

Syntax

```
BOOL GetCustomVideoTimingsCount(  
    DWORD *          pcTimings  
);
```

Params

pcTimings [out]

The number of custom video timings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.63 GetCustomVideoTimingsArray

The method can be used to get custom video all timings.

Syntax

```
BOOL GetCustomVideoTimingsArray(  
    MWCAP_VIDEO_CUSTOM_TIMING *    pTiming,  
    DWORD * pcTimings
```

```
);
```

Params

pTiming [out]

The interface of custom video timings.

pcTimings [out]

Return the number of custom video timings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_CUSTOM_TIMING structure.

3.1.64 SetCustomVideoTimingsArray

The method can be used to set custom video all timings.

Syntax

```
BOOL SetCustomVideoTimingsArray(  
    const MWCAP_VIDEO_CUSTOM_TIMING *    pTiming,  
    DWORD cTimings  
);
```

Params

pTiming [in]

The interface of custom video timings.

cTimings [in]

Set the number of custom video timings.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

Reference the description of MWCAP_VIDEO_CUSTOM_TIMINGstructure.

3.1.65 GetCustomVideoResolutionsCount

The method can be used to get custom video all resolutions count.

Syntax

```
BOOL GetCustomVideoResolutionsCount(  
    DWORD *    pcResolutions  
);
```

Params

pcResolutions [out]

The number of custom video resolutions.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.66 GetCustomVideoResolutionsArray

The method can be used to get audio channel all input sources.

Syntax

```
BOOL GetCustomVideoResolutionsArray(  
    SIZE *    pResolutions,  
    DWORD *    pcResolutions  
);
```

Params

pResolutions [out]

The interface of custom video resolutions.

pcResolutions [out]

Return the number of custom video resolutions.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.67 SetCustomVideoResolutionsArray

The method can be used to set custom video all resolutions.

Syntax

```
BOOL SetCustomVideoResolutionsArray(  
    const SIZE *    pResolutions,  
    DWORD          cResolutions  
);
```

Params

pResolutions [in]

The interface of custom video resolutions.

cResolutions [in]

Set the number of custom video resolutions.

Return Value

Return HRESULT value. Possible values include the following.

S_OK	Success.
S_FALSE	Fail.

Remark

None

3.1.68 GetCoreTemperature

The method can be used to get the current temperature value of the card.

Syntax

```
void GetCoreTemperature (  
    LONG *    pTemperatureDegC_X10  
);
```

Params

pTemperatureDegC_X10[out]

Retrun the temperature value.

Return Value

None

Remark

None

3.2 MWCapture Interface Functions

3.2.1 MWGetVersion method

The method returns the version of Pro_Device interface.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWGetVersion(
BYTE*	pbyMaj,
BYTE*	pbyMin,
WORD*	pwBuild
);	

Params

- pbyMaj [out]
Major version number.
- pbyMin [out]
Minor version number.
- pwBuild [out]
Build version number.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

This function always returns MW _SUCCEEDED. If parameters are invalid, they will not be filled

3.2.2 MWCaptureInitInstance method

The method can be used toInitialize MWCapture library.

Syntax

BOOL
LIBMWCAPTURE_API
MWCaptureInitInstance(
);

Params

None.

Return Value

Return BOOL value. Possible values include the following.

TRUE	Success.
FALSE	Fail.

Remark

None

3.2.3 MWCaptureExitInstance method

The method can be used to Close the MWCapture library, free all resources that Dll remains.

Syntax

```
void  
LIBMWCAPTURE_API  
MWCaptureExitInstance(  
    );
```

Params

None.

Return Value

None

Remark

None.

3.2.4 MWRefreshDevice method

The method can be used to re-enumerates capture devices on computer.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWRefreshDevice(  
    );
```

Params

None.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.5 MWGetChannelCount method

The method returns total channels count.

Syntax

```
int
LIBMWCAPTURE_API
MWGetChannelCount(
    );
```

Params

None.

Return Value

Returns channel count.

Remark

None

3.2.6 MWGetChannelInfoByIndex method

The method can be used to get channel information by index

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetChannelInfoByIndex(
    int                                nIndex,
    MWCAP_CHANNEL_INFO *              pChannelInfo
    );
```

Params

nIndex [in]

Channel index which ranges from 0 to MWGetChannelCount() – 1.

pChannelInfo [out]

Receives channel information.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.7 MWGetFamilyInfoByIndex method

The method can be used to get family information of capture device by index.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetFamilyInfoByIndex(  
    int                                nIndex,  
    LPVOID                            pFamilyInfo,  
    DWORD                             dwSize  
);
```

Params

nIndex [in]

Channel index which ranges from 0 to MWGetChannelCount() – 1.

pFamilyInfo [out]

Pointer to struct MWCAP_PRO_CAPTURE_INFO. Receives family information.

dwSize [in]

Size of struct MWCAP_PRO_CAPTURE_INFO.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.8 MWGetDevicePath method

The method can be used to get instance path of capture device by index.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetDevicePath(  
    int                                nIndex,  
    WCHAR*                            pDevicePath  
);
```

Params

nIndex [in]

Channel index which ranges from 0 to MWGetChannelCount() – 1.

pDevicePath [out]

Receives device instance path.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.9 MWOpenChannel method

The method can be used to open channel by rotary switch value on the board and channel index.

Syntax

```
HCHANNEL  
LIBMWCAPTURE_API  
MWOpenChannel(  
    int                                nBoardValue,  
    int                                nChannelIndex  
);
```

Params

nBoardValue [in]

The value of rotary switch on the board.

nChannelIndex [in]

Channel index which ranges from 0.

Return Value

Returns channel handle if success. Otherwise return NULL

Remark

If two capture devices have the same switch value, the first one on PCI-e slots will be opened. A channel can be opened multiple times. The handles that got by opening the same channel are independent

3.2.10 MWOpenChannelByPath method

The method can be used to open channel by device instance path.

Syntax

```
HCHANNEL  
LIBMWCAPTURE_API  
MWOpenChannelByPath(  
    const WCHAR*                pszDevicePath  
);
```

Params

pszDevicePath [in]
Device instance path.

Return Value

Returns channel handle if success. Otherwise return NULL.

Remark

None

3.2.11 MWCloseChannel method

The method can be used to close the channel.

Syntax

```
void  
LIBMWCAPTURE_API  
MWCloseChannel(  
    HCHANNEL                    hChannel  
);
```

Params

hChannel [in]
Input the channel handle that had been opened.

Return Value

None

Remark

None

3.2.12 MWGetChannelInfo method

The method can be used to get the opened channel information..

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetChannelInfo(  
    HCHANNEL                hChannel,  
    MWCAP_CHANNEL_INFO *    pChannelInfo  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pChannelInfo [out]

Receives channel information.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.13 MWGetFamilyInfo method

The method can be used to get family information of capture device.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetFamilyInfo(  
    HCHANNEL                hChannel,  
    LPVOID                  pFamilyInfo,  
    DWORD                   dwSize  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pFamilyInfo [out]

Receives family information of capture device that the channel belongs to.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.14 MWGetVideoCaps method

The method can be used to get video capturing capacity of the channel.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetVideoCaps(  
    HCHANNEL                hChannel,  
    MWCAP_VIDEO_CAPS*       pVideoCaps  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pVideoCaps [out]

Receives video capturing capacity.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

Receives video input sources.

pdwInputCount[out]

Receives count of video input sources.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.17 MWGetAudioInputSourceArray method

The method can be used to get audio input source information of the channel.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWGetAudioInputSourceArray(
HCHANNEL	hChannel,
DWORD*	pdwInputSource,
DWORD*	pdwInputCount
);	

Params

hChannel [in]

Input the channel handle that had been opened.

pdwInputSource [out]

Receives audio input sources.

pdwInputCount[out]

Receives count of audio input sources.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.18 MWGetInputSourceScan method

The method can be used to get scanning state of input source.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetInputSourceScan(  
    HCHANNEL                hChannel,  
    BOOLEAN*                pbScan  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pbScan [out]

Receives whether is scanning..

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.19 MWSetInputSourceScan method

The method can be used to set scanning state of input source.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWSetInputSourceScan(  
    HCHANNEL                hChannel,  
    BOOLEAN                bScan  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

bScan [in]

Set whether is scanning.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.20 MWGetAVInputSourceLink method

The method can be used to get link state of input source.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetAVInputSourceLink(  
    HCHANNEL                hChannel,  
    BOOLEAN*                pbLink  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pbLink [out]

Get whether is linked.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.21 MWSetAVInputSourceLink method

The method can be used to set link state of input source.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API
```

```

MWSetAVInputSourceLink(
    HCHANNEL                hChannel,
    BOOLEAN                  bLink
);

```

Params

hChannel [in]

Input the channel handle that had been opened.

bLink [in]

Set whether is linked.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.22 MWGetVideoInputSource method

The method can be used to get the current video input source of the channel.

Syntax

```

MW_RESULT
LIBMWCAPTURE_API
MWGetVideoInputSource(
    HCHANNEL                hChannel,
    DWORD*                  pdwSource
);

```

Params

hChannel [in]

Input the channel handle that had been opened.

pdwSource [out]

Receives the current video input source.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.23 MWSetVideoInputSource method

The method can be used to set the current video input source of the channel.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWSetVideoInputSource(  
    HCHANNEL                hChannel,  
    DWORD                   dwSource  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

dwSource [in]

Set the current video input source.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.24 MWGetAudioInputSource method

The method can be used to get the current audio input source of the channel.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetAudioInputSource(  
    HCHANNEL                hChannel,  
    DWORD*                  pdwSource  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pdwSource [out]

Receives the current audio input source.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.25 MWSetAudioInputSource method

The method can be used to set the current audio input source of the channel.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWSetAudioInputSource(  
    HCHANNEL                hChannel,  
    DWORD                   dwSource  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

dwSource [in]

Set the current audio input source.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.26 MWGetEDID method

The method can be used to get EDID data of the channel.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetEDID(
    HCHANNEL                hChannel,
    BYTE*                   pbyData,
    ULONG*                  pulSize
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pbyData [out]

Receives EDID data.

pulSize [out]

Receives EDID data length.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.27 MWSetEDID method

The method can be used to set EDID data of the channel.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWSetEDID(
    HCHANNEL                hChannel,
    BYTE*                   pbyData,
    ULONG                   ulSize
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pbyData [in]

New EDID data.
ulSize [in]
EDID data length.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.28 MWGetInputSpecificStatus method

The method can be used to get specific input signal status of the channel.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetInputSpecificStatus(  
    HCHANNEL                hChannel,  
    MWCAP_INPUT_SPECIFIC_STATUS * pInputStatus  
);
```

Params

hChannel [in]
Input the channel handle that had been opened.
pInputStatus [out]
Receives specific input signal status

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.29 MWGetVideoSignalStatus method

The method can be used to get video signal status of the channel.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetVideoSignalStatus(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_SIGNAL_STATUS * pSignalStatus
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pSignalStatus [out]

Receives video signal status

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.30 MWGetAudioSignalStatus method

The method can be used to get audio signal status of the channel.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetAudioSignalStatus(
    HCHANNEL                hChannel,
    MWCAP_AUDIO_SIGNAL_STATUS * pSignalStatus
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pSignalStatus [out]

Receives audio signal status

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.31 MWGetHDMIInfoFrameValidFlag method

The method can be used to get the valid flag in HDMI info frame.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetHDMIInfoFrameValidFlag(
    HCHANNEL                hChannel,
    DWORD*                  pdwValidFlag
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pdwValidFlag [out]

Receives the valid flag in HDMI info frame

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.32 MWGetHDMIInfoFramePacket method

The method can be used to get packet in HDMI info frame.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetHDMIInfoFramePacket(
    HCHANNEL                hChannel,
    MWCAP_HDMI_INFOFRAME_ID id,
```

HDMI_INFOFRAME_PACKET*	pPacket
);	

Params

hChannel [in]

Input the channel handle that had been opened.

id [in]

ID of the HDMI info frame

pPacket [out]

Recieves packet in HDMI info frame.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.33 MWGetDeviceTime method

The method can be used to get current time from capture device.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWGetDeviceTime(
HCHANNEL	hChannel,
LONGLONG*	pllTime
);	

Params

hChannel [in]

Input the channel handle that hadbeen opened.

pllTime[out]

Receives the time from device in 100 nanosecond units.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

The different channels in one device used the uniform clock. It is very important for video and audio synchronization, or multi-channels synchronization.

3.2.34 MWSetDeviceTime method

The method can be used to set current time on capture device.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWSetDeviceTime(
    HCHANNEL                hChannel,
    LONGLONG                 lTime
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

lTime[in]

Set the time on device. It is in 100 nanosecond units.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

This function can be used to synchronous clocks on different devices

3.2.35 MWRegulateDeviceTime method

The method can be used to regulate current time in capture device.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWRegulateDeviceTime(
    HCHANNEL                hChannel,
    LONGLONG                 lTime
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

lTime[in]

Regulate the time to device.It is in 100 nanosecond units.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

The method can be used to regulate current time in capture device.

Different with MWSetDeviceTime, after setting, the time on device will not change to the input value immediately. It will regulate gradually. While after setting by MWSetDeviceTime, the time on device will change to the input value immediately

3.2.36 MWRegisterTimer method

The method can be used to register timer event.

Syntax

```
HTIMER
LIBMWCAPTURE_API
MWRegisterTimer(
    HCHANNEL          hChannel,
    HANDLE             hEvent
);
```

Params

hChannel [in]

Input the channel handle that hadbeen opened.

hEvent [in]

An event Handle for timer event. It need be created by CreateEvent() (Kernel32 Function).

Return Value

Returns the handle of timer event. If error occur, it will return NULL .

Remark

None.

Input the handle of timer event that be created by MWRegisterTimerEvent() .

llExpireTime [in]

Input the expire time on device clock. And it is the absolute time.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

When the schedule time is expired, the event handle related by MWRegisterTimerEvent() will be set to the signaled state.

3.2.39 MWRegisterNotify method

The method can be used to register notify event object .

Syntax

```
HNOTIFY
LIBMWCAPTURE_API
MWRegisterNotify(
    HCHANNEL          hChannel,
    HANDLE             hEvent,
    DWORD              dwEnableBits
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

hNotifyEvent[in]

An event Handle for timer event. It need be created by CreateEvent() (Kernel32 Function) .

dwEnableBits[in]

The mask of bits for notify event. See Remark for more details.

Return Value

Returns the handle of notify event. If error occur, it will return NULL .

Remark

dwEnableBits can be set in follow mask value to enable notify.

MWCAP_NOTIFY_INPUT_SOURCE_CHANGE	The input signal changed.
MWCAP_NOTIFY_INPUT_FORMAT_CHANGE	The input signal format changed.
MWCAP_NOTIFY_VIDEO_FIELD_INPUT	The input signal state changed.
MWCAP_NOTIFY_VIDEO_FRAME_BUFFERED	The video frame buffered.

MWCAP_NOTIFY_VIDEO_INPUT_RESET | The capture operation had been reset.
When the relation notify occur, hEvent will be set to the signal state.

3.2.40 MWUnregisterNotify method

The method can be used to unregister notify event object .

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWUnregisterNotify(  
    HCHANNEL                hChannel,  
    HNOTIFY                 hNotify  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

hNotify[in]

Input the handle of notify event that be created by MWRegisterNotify().

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.41 MWGetNotifyStatus method

The method can be used to get notify object status.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetNotifyStatus(  
    HCHANNEL                hChannel,  
    HNOTIFY                 hNotify,  
    ULONGLONG*              pullStatus  
);
```

Params

hChannel [in]

Input the channel handle that had opened.

hNotify[in]

Input the handle of notify event that be created by MWRegisterNotify() .

pullStatus [out]

Get current status mask that changed.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.42 MWStartVideoCapture method

Start capture channel to getvideo frames.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWStartVideoCapture(
    HCHANNEL          hChannel,
    HANDLE             hEvent
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

hEvent[in]

Set event handle to get asynchronous notify. It need be created by CreateEvent() (Kernel32 Function)

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.43 MWStopVideoCapture method

Stop video capturing.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWStopVideoCapture(  
    HCHANNEL                hChannel  
);
```

Params

hChannel [in]

Input the channel handle had started video capturing by MWStartVideoCapture.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.44 MWPinVideoBuffer method

Pin a piece of virtual memory.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWPinVideoBuffer(  
    HCHANNEL                hChannel,  
    LPBYTE                  pbFrame,  
    DWORD                   cbFrame  
);
```

Params

hChannel [in]

Input the channel handle had started video capturing by MWStartVideoCapture.

pbFrame [in]

the virtual memory address.

cbFrame [in]

the size of the virtual memory.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

.

3.2.45 MWUnpinVideoBuffer method

Unpin a piece of virtual memory.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWUnpinVideoBuffer(  
    HCHANNEL                hChannel,  
    LPBYTE                  pbFrame  
);
```

Params

hChannel [in]

Input the channel handle had started video capturing by MWStartVideoCapture.

pbFrame [in]

the virtual memory address.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.46 MWCaptureVideoFrameToVirtualAddress method

Capture video frame data to virtual memory address.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API
```

```

MWCaptureVideoFrameToVirtualAddress(
    HCHANNEL                hChannel,
    int                      iFrame,
    LPBYTE                   pbFrame,
    DWORD                    cbFrame,
    DWORD                    cbStride,
    BOOLEAN                  bBottomUp,
    MWCAP_PTR64              pvContext,
    DWORD                    dwFOURCC,
    int                      cx,
    int                      cy
);

```

Params

hChannel [in]

Input the channel handle that had started video capturing.

iFrame[in]

Set index of frame to capture

pbFrame [out]

The buffer point of virtual memory address for filling the capture frame.

cbFrame[in]

pvFrame length.

cbStride[in]

The stride of pvFrame.

bBottomUp[in]

Indicate the capture frame is bottom-up or not.

pvContext[in]

A custom content point.

dwFOURCC[in]

Indicate the capture frame color format. Reference to MWFOURCC.h.

cx [in]

Indicate the capture frame width.

cy [in]

Indicate the capture frame height.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.47 MWCaptureVideoFrameToPhysicalAddress method

Capture channel frame by physics memory address.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWCaptureVideoFrameToPhysicalAddress(
    HCHANNEL                hChannel,
    int                      iFrame,
    LONGLONG                lFrameAddress,
    DWORD                   cbFrame,
    DWORD                   cbStride,
    BOOLEAN                 bBottomUp,
    MWCAP_PTR64             pvContext,
    DWORD                   dwFOURCC,
    int                     cx,
    int                     cy
);
```

Params

hChannel [in]

Input the channel handle that had started video capturing.

iFrame[in]

Set index of frame to capture

lFrameAddress[in]

The physical memory address for filling the capture frame.

cbFrame[in]

pvFrame length.

cbStride[in]

The stride of pvFrame.

bBottomUp[in]

Indicate the capture frame is bottom-up or not.

pvContext[in]

A custom content point.

dwFOURCC[in]

Indicate the capture frame color format. Reference to MWFOURCC.h.

cx [in]

Indicate the capture frame width.

cy [in]

Indicate the capture frame height.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.48 MWCaptureVideoFrameWithOSDToVirtualAddress method

Capture video frame data to virtual memory address.

Syntax

```

MW_RESULT
LIBMWCAPTURE_API
MWCaptureVideoFrameWithOSDToVirtualAddress(
    HCHANNEL                hChannel,
    int                      iFrame,
    LPBYTE                   pbFrame,
    DWORD                    cbFrame,
    DWORD                    cbStride,
    BOOLEAN                  bBottomUp,
    MWCAP_PTR64              pvContext,
    DWORD                    dwFOURCC,
    int                      cx,
    int                      cy,
    HOSD                     hOSDImage,
    const RECT *             pOSDRects,
    int                      cOSDRects
);

```

Params

hChannel [in]

Input the channel handle that had started video capturing.

iFrame[in]

Set index of frame to capture

pbFrame [out]

The buffer point of virtual memory address for filling the capture frame.

cbFrame[in]

pvFrame length.

cbStride[in]

The stride of pvFrame.

bBottomUp[in]

Indicate the capture frame is bottom-up or not.

pvContext[in]

A custom content point.

dwFOURCC[in]

Indicate the capture frame color format. Reference to MWFOURCC.h.

cx [in]

Indicate the capture frame width.

cy [in]

Indicate the capture frame height.

hOSDImage [in]

OSD image handle got by invoking MWCreateImage.

pOSDRects[in]

Target areas to blend OSD image.

cOSDRects[in]

Count of arget areas to blend OSD image.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.49 MWCaptureVideoFrameWithOSDToPhysicalAddress method

Capture channel frame by physics memory address.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWCaptureVideoFrameWithOSDToPhysicalAddress(
HCHANNEL	hChannel,
int	iFrame,
LONGLONG	lFrameAddress,
DWORD	cbFrame,
DWORD	cbStride,
BOOLEAN	bBottomUp,
MWCAP_PTR64	pvContext,
DWORD	dwFOURCC,
int	cx,
int	cy,
HOSD	hOSDImage,
const RECT *	pOSDRects,
int	cOSDRects

```
);
```

Params

hChannel [in]

Input the channel handle that had started video capturing.

iFrame[in]

Set index of frame to capture

llFrameAddress[in]

The physical memory address for filling the capture frame.

cbFrame[in]

pvFrame length.

cbStride[in]

The stride of pvFrame.

bBottomUp[in]

Indicate the capture frame is bottom-up or not.

pvContext[in]

A custom content point.

dwFOURCC[in]

Indicate the capture frame color format. Reference to MWFOURCC.h.

cx [in]

Indicate the capture frame width.

cy [in]

Indicate the capture frame height.

hOSDImage[in]

OSD image handle got by invoking MWCreateImage.

pOSDRects[in]

Target areas to blend OSD image.

cOSDRects[in]

Count of arget areas to blend OSD image.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.50 MWCaptureVideoFrameToVirtualAddressEx method

Capture video frame data to virtual memory address.

Syntax


```

MW_RESULT
LIBMWCAPTURE_API
MWCaptureVideoFrameToVirtualAddressEx(
    HCHANNEL                hChannel,
    int                     iFrame,
    LPBYTE                  pbFrame,
    DWORD                   cbFrame,
    DWORD                   cbStride,
    BOOLEAN                  bBottomUp,
    MWCAP_PTR64             pvContext,
    DWORD                   dwFOURCC,
    int                     cx,
    int                     cy,
    DWORD                   dwProcessSwitchs,
    int                     cyParitalNotify,
    HOSD                    hOSDImage,
    const RECT *             pOSDRects,
    int                     cOSDRects,
    SHORT                   sContrast,
    SHORT                   sBrightness,
    SHORT                   sSaturation,
    SHORT                   sHue,
    MWCAP_VIDEO_DEINTERLACE_MODE    deinterlaceMode,
    MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODE    aspectRatioConvertMode,
    const RECT *             pRectSrc,
    const RECT *             pRectDest,
    int                     nAspectX,
    int                     nAspectY,
    MWCAP_VIDEO_COLOR_FORMAT    colorFormat,
    MWCAP_VIDEO_QUANTIZATION_RANGE    quantRange,
    MWCAP_VIDEO_SATURATION_RANGE    satRange
);

```

Params

hChannel [in]

Input the channel handle that had started video capturing.

iFrame[in]

Set index of frame to capture

pbFrame [out]

The buffer point of virtual memory address for filling the capture frame.

cbFrame[in]

pvFrame length.

cbStride[in]

The stride of pvFrame.

bBottomUp[in]
Indicate the capture frame is bottom-up or not.

pvContext[in]
A custom content point.

dwFOURCC[in]
Indicate the capture frame color format. Reference to MWFOURCC.h.

cx [in]
Indicate the capture frame width.

cy [in]
Indicate the capture frame height.

dwProcessSwitchs[in]
Mask of video processes. Reference to MWCAP_VIDEO_PROCESS_xx.

cyPartialNotify[in]
Count of partial notifies.

pOSDImage[in]
OSD image handle got by invoking MWCreateImage.

pOSDRects[in]
Target areas to blend OSD image.

cOSDRects[in]
Count of arget areas to blend OSD image.

sContrast[in]
Contrast.

sBrightness[in]
Brightness.

sSaturation[in]
Saturation.

sHue[in]
Hue.

deinterlaceMode[in]
De-interlace mode.

aspectRatioConvertMode[in]
Aspect ratio convert mode.

pRectSrc [in]
Source area of image to capture.

pRectDest[in]
Destination area of image to capture.

nAspectX[in]
Width in aspect ratio.

nAspectY[in]
Height in aspect ratio.

colorFormat[in]
Color format standard.

quantRange[in]
Quantization range.

satRange[in]

Saturation range.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.51 MWCaptureVideoFrameToPhysicalAddressEx method

Capture video frame data to physical memory address.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWCaptureVideoFrameToPhysicalAddressEx(
HCHANNEL	hChannel,
int	iFrame,
LONGLONG	llFrameAddress,
DWORD	cbFrame,
DWORD	cbStride,
BOOLEAN	bBottomUp,
MWCAP_PTR64	pvContext,
DWORD	dwFOURCC,
int	cx,
int	cy,
DWORD	dwProcessSwitchs,
int	cyParitalNotify,
HOSD	hOSDImage,
const RECT *	pOSDRects,
int	cOSDRects,
SHORT	sContrast,
SHORT	sBrightness,
SHORT	sSaturation,
SHORT	sHue,
MWCAP_VIDEO_DEINTERLACE_MODE	deinterlaceMode,
MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODE	aspectRatioConvertMode,
const RECT *	pRectSrc,
const RECT *	pRectDest,
int	nAspectX,

int	nAspectY,
MWCAP_VIDEO_COLOR_FORMAT	colorFormat,
MWCAP_VIDEO_QUANTIZATION_RANGE	quantRange,
MWCAP_VIDEO_SATURATION_RANGE	satRange
);	

Params

hChannel [in]

Input the channel handle that had started video capturing.

iFrame[in]

Set index of frame to capture

llFrameAddress[in]

The physical memory address for filling the capture frame.

cbFrame[in]

pvFrame length.

cbStride[in]

The stride of pvFrame.

bBottomUp[in]

Indicate the capture frame is bottom-up or not.

pvContext[in]

A custom content point.

dwFOURCC[in]

Indicate the capture frame color format. Reference to MWFOURCC.h.

cx [in]

Indicate the capture frame width.

cy [in]

Indicate the capture frame height.

dwProcessSwitchs[in]

Mask of video processes. Reference to MWCAP_VIDEO_PROCESS_xx.

cyPartialNotify[in]

Count of partial notifies.

pOSDImage[in]

OSD image handle got by invoking MWCreateImage.

pOSDRects[in]

Target areas to blend OSD image.

cOSDRects[in]

Count of arget areas to blend OSD image.

sContrast[in]

Contrast.

sBrightness[in]

Brightness.

sSaturation[in]

Saturation.

sHue[in]

Hue.

deinterlaceMode[in]
De-interlace mode.

aspectRatioConvertMode[in]
Aspect ratio convert mode.

pRectSrc [in]
Source area of image to capture.

pRectDest[in]
Destination area of image to capture.

nAspectX[in]
Width in aspect ratio.

nAspectY[in]
Height in aspect ratio.

colorFormat[in]
Color format standard.

quantRange[in]
Quantization range.

satRange[in]
Saturation range.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.52 MWGetVideoBufferInfo method

Get video buffering information.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetVideoBufferInfo(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_BUFFER_INFO * pVideoBufferInfo
);
```

Params

hChannel [in]
Input the channel handle that had started video capturing.

pVideoBufferInfo [out]
Video buffering information.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.53 MWGetVideoFrameInfo method

Get video frame information.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetVideoFrameInfo(  
    HCHANNEL                hChannel,  
    BYTE                    i,  
    MWCAP_VIDEO_FRAME_INFO* pVideoFrameInfo  
);
```

Params

hChannel [in]
Input the channel handle that had started video capturing.

i [in]
Index of video frame.

pVideoFrameInfo [out]
Receives video frame information.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.54 MWGetVideoCaptureStatus method

Get video capturing status.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetVideoCaptureStatus(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_CAPTURE_STATUS * pStatus
);
```

Params

hChannel [in]

Input the channel handle that had started video capturing.

pStatus [out]

Receives video capturing status.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.55 MWStartAudioCapture method

Start audio capturing in this channel.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWStartAudioCapture(
    HCHANNEL                hChannel
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.

MW_INVALID_PARAMS | Input param error.

Remark

None.

3.2.56 MWStopAudioCapture method

Stop audio capturing in this channel.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWStopAudioCapture(
    HCHANNEL                hChannel
);
```

Params

hChannel [in]
Input the channel handle that had started capturing audio.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.57 MWCaptureAudioFrame method

Capture an audio frame.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWCaptureAudioFrame(
    HCHANNEL                hChannel,
    MWCAP_AUDIO_CAPTURE_FRAME* pAudioCaptureFrame
);
```

Params

hChannel [in]
Input the channel handle that had started capturing audio.

pAudioCaptureFrame [out]
Receives an audio frame.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.58 MWSetVideoInputAspectRatio method

Set the aspect ratio of input videosignal.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWSetVideoInputAspectRatio(
HCHANNEL	hChannel,
int	nAspectX,
int	nAspectY
);	

Params

hChannel [in]
Input the channel handle that had been opened.

nAspectX [in]
Set width in aspect ratio.

nAspectY [in]
Set height in aspect ratio.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.59 MWGetVideoInputAspectRatio method

Get the aspect ratio of input videosignal.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetVideoInputAspectRatio(
    HCHANNEL                hChannel,
    int*                     pnAspectX,
    int*                     pnAspectY
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pnAspectX [out]

Receives width in aspect ratio.

pnAspectY [out]

Receives height in aspect ratio.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.60 MWSetVideoInputColorFormat method

Set video input color space standard.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWSetVideoInputColorFormat(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_COLOR_FORMAT colorFormat
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

colorFormat [in]

Color format standard.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.61 MWGetVideoInputColorFormat method

Get video input color space standard.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWGetVideoInputColorFormat(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_COLOR_FORMAT * pColorFormat
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pColorFormat [out]

Receives color format standard.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.62 MWSetVideoInputQuantizationRange method

Set video input quantization range.

Syntax

```
MW_RESULT
```

```

LIBMWCAPTURE_API
MWSetVideoInputQuantizationRange(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_QUANTIZATION_RANGE  quantRange
);

```

Params

hChannel [in]

Input the channel handle that had been opened.

quantRange [in]

Quantization range

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.63 MWGetVideoInputQuantizationRange method

Get video input quantization range.

Syntax

```

MW_RESULT
LIBMWCAPTURE_API
MWGetVideoInputQuantizationRange(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_QUANTIZATION_RANGE*  pQuantRange
);

```

Params

hChannel [in]

Input the channel handle that had been opened.

pQuantRange [in]

Receives quantization range

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.64 MWSetLEDMode method

Set LED mod.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWSetLEDMode(  
    HCHANNEL                hChannel,  
    DWORD                   dwMode  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

dwMode [in]

LED mode. Reference to MWCAP_LED_MODE.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None.

3.2.65 MWGetFirmwareStorageInfo method

Get firmware storage infomation.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWGetFirmwareStorageInfo(  
    HCHANNEL                hChannel,  
    MWCAP_FIRMWARE_STORAGE * pFirmwareStorageInfo  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

pFirmwareStorageInfo [out]

Receives firmware storage information.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.66 MWEraseFirmwareData method

Erase firmware data.

Syntax

```
MW_RESULT  
LIBMWCAPTURE_API  
MWEraseFirmwareData(  
    HCHANNEL                hChannel,  
    DWORD                   cbOffset,  
    DWORD                   cbErase  
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

cbOffset [in]

The start position of data to be erased.

cbErase [in]

The length of data to be erased.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.67 MWReadFirmwareData method

Read firmware data.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWReadFirmwareData(
HCHANNEL	hChannel,
DWORD	cbOffset,
BYTE *	pbyData,
DWORD	cbToRead,
DWORD *	pcbRead
);	

Params

hChannel [in]

Input the channel handle that had been opened.

cbOffset [in]

The start position of data to be erased.

pbyData [out]

Receives data readed.

cbToRead [in]

Desired data length to read.

pcbRead [out]

Receives the actual read length.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.68 MWWriteFirmwareData method

Write firmware data.

Syntax

MW_RESULT
LIBMWCAPTURE_API
MWWriteFirmwareData(

HCHANNEL	hChannel,
DWORD	cbOffset,
BYTE *	pbyData,
DWORD	cbData
);	

Params

hChannel [in]

Input the channel handle that had been opened.

cbOffset [in]

The start position of data to be erased.

pbyData [in]

Data to write.

cbData [in]

Length of data to write.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.69 MWSetPostReconfig method

Post reconfig command to execute after a while.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWSetPostReconfig(
HCHANNEL	hChannel,
DWORD	dwDelayMS
);	

Params

hChannel [in]

Input the channel handle that had been opened.

dwDelayMS [in]

The delay to execute reconfig in milliseconds.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.70 MWCreateImage method

Create OSD image. .

Syntax

```
HOSD
LIBMWCAPTURE_API
MWCreateImage(
    HCHANNEL          hChannel,
    int               cx,
    int               cy
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

cx [in]

Width of image.

cy [in]

Height of image.

Return Value

Returns handle of this image. If error occurs, return NULL.

Remark

None

3.2.71 MWOpenImage method

Open image.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWOpenImage(
    HCHANNEL          hChannel,
```

HOSD LONG*);	hImage, plRet
---------------------	------------------

Params

hChannel [in]

Input the channel handle that had been opened.

hImage [in]

Handle of image.

plRet [out]

Receives the count of references to this image.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.72 MWCloseImage method

Close image.

Syntax

MW_RESULT LIBMWCAPTURE_API MWCloseImage(HCHANNEL HOSD LONG*);	hChannel, hImage, plRet
---	-------------------------------

Params

hChannel [in]

Input the channel handle that had been opened.

hImage [in]

Handle of image.

plRet [out]

Receives the count of references to this image.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.73 MWUploadImageFromVirtualAddress method

Upload image from system memory to capture device.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWUploadImageFromVirtualAddress(
    HCHANNEL                hChannel,
    HOSD                    hImage,
    MWCAP_VIDEO_COLOR_FORMAT cfDest,
    MWCAP_VIDEO_QUANTIZATION_RANGE quantRangeDest,
    MWCAP_VIDEO_SATURATION_RANGE satRangeDest,
    WORD                    xDest,
    WORD                    yDest,
    WORD                    cxDest,
    WORD                    cyDest,
    MWCAP_PTR64             pvSrcFrame,
    DWORD                   cbSrcFrame,
    DWORD                   cbSrcStride,
    WORD                    cxSrc,
    WORD                    cySrc,
    BOOLEAN                 bSrcBottomUp,
    BOOLEAN                 bSrcPixelAlpha,
    BOOLEAN                 bSrcPixelXBGR
);
```

Params

hChannel [in]

Input the channel handle that had been opened.

hImage [in]

Handle of image.

cfDest [in]

Color format standard of the target image.

quantRangeDest [in]

Quantization rage of the target image.

satRangeDest [in]

Saturation range of the target image.

xDest [in]
Target position in horizontal direction.

yDest [in]
Target position in vertical direction.

cxDest [in]
Width of target image.

cyDest [in]
Height of target image.

pvSrcFrame [in]
Source image data.

cbSrcFrame [in]
Length of source image data.

cbSrcStride [in]
Stride of source image data.

cxSrc [in]
Width of source image.

cySrc [in]
Height of source image.

bSrcBottomUp [in]
Whether source image is bottom up..

bSrcPixelAlpha [in]
Whether source image pixels have alpha component..

bSrcPixelXBGR [in]
Whether color format of source image pixels is XBGR..

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.74 MWUploadImageFromPhysicalAddress method

Upload image from physical memory to capture device.

Syntax

```
MW_RESULT
LIBMWCAPTURE_API
MWUploadImageFromPhysicalAddress(
    HCHANNEL                                hChannel,
```

HOSD	hImage,
MWCAP_VIDEO_COLOR_FORMAT	cfDest,
MWCAP_VIDEO_QUANTIZATION_RANGE	quantRangeDest,
MWCAP_VIDEO_SATURATION_RANGE	satRangeDest,
WORD	xDest,
WORD	yDest,
WORD	cxDest,
WORD	cyDest,
LONGLONG	lSrcFrameAddress,
DWORD	cbSrcFrame,
DWORD	cbSrcStride,
WORD	cxSrc,
WORD	cySrc,
BOOLEAN	bSrcBottomUp,
BOOLEAN	bSrcPixelAlpha,
BOOLEAN	bSrcPixelXBGR
);	

Params

hChannel [in]

Input the channel handle that had been opened.

hImage [in]

Handle of image.

cfDest [in]

Color format standard of the target image.

quantRangeDest [in]

Quantization rage of the target image.

satRangeDest [in]

Saturation range of the target image.

xDest [in]

Target position in horizontal direction.

yDest [in]

Target position in vertical direction.

cxDest [in]

Width of target image.

cyDest [in]

Height of target image.

lSrcFrameAddress [in]

Physical address that source image data is stored in.

cbSrcFrame [in]

Length of source image data.

cbSrcStride [in]

Stride of source image data.

cxSrc [in]

Width of source image.

cySrc [in]
Height of source image.

bSrcBottomUp [in]
Whether source image is bottom up..

bSrcPixelAlpha [in]
Whether source image pixels have alpha component..

bSrcPixelXBGR [in]
Whether color format of source image pixels is XBGR..

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.
MW_INVALID_PARAMS	Input param error.

Remark

None

3.2.75 MWCreateExtendObject method

The method can be used to get the channel expand information.

Syntax

MW_RESULT	
LIBMWCAPTURE_API	
MWCreateExtendObject(
HCHANNEL	hChannel,
GUID	guid,
LPVOID*	ppObject
);	

Params

hChannel [in]
Input the channel handle that had been opened.

guid [in]
expand information flags.

ppObject [out]
Receives the expand information.

Return Value

Returns an MW_RESULT value. Possible values include the following.

MW_SUCCEEDED	Success.
MW_FAILED	Fail.

MW_INVALID_PARAMS | Input param error.

Remark

None

4 Contact Us

Please access the follow web to get the least version.

<http://www.magewell.com/sdk>

If any questions, please send mail to support@magewell.net .