

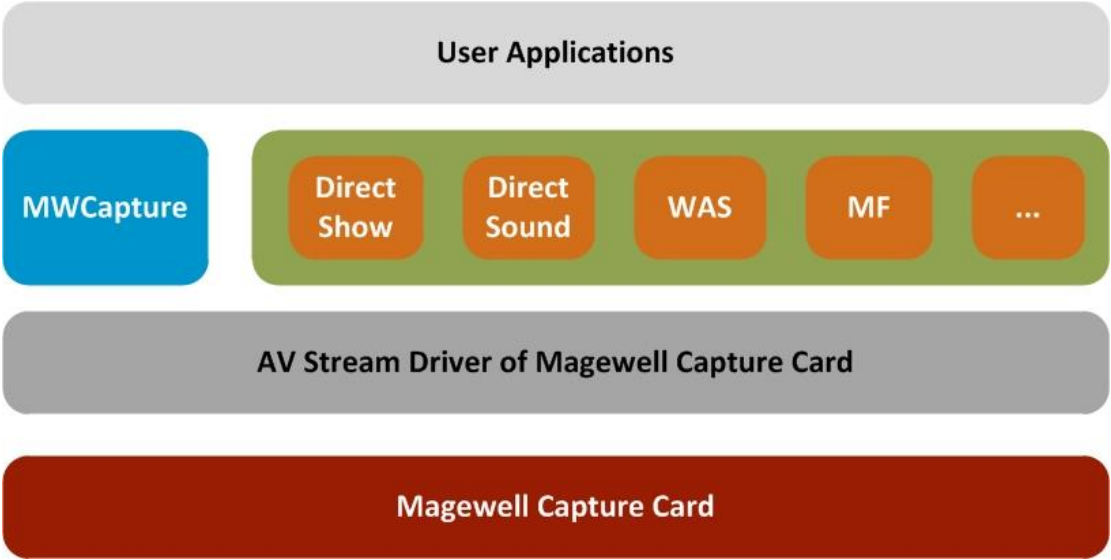
Pro Capture 系列采集卡

软件开发手册

1 简介.....	2
2 类型定义.....	3
2.1 枚举类型	3
2.2 数据结构	17
3 函数说明.....	45
3.1 Direct Show 扩展接口函数	45
3.2MWCapture 接口函数	80
4 联系我们.....	135

1 简介

Pro Capture 软件开发工具包主要用于开发 Pro Capture 系列采集卡的应用软件。它直接通过 MWCapture 访问 Pro Capture 系列采集卡的驱动程序，基于它来开发应用软件可以实现 Pro Capture 系列采集卡的全部功能特性，并且可以获得最佳的性能和更好的灵活性。同时我们也提供 DirectShow 扩展接口，方便基于 DirectShow 开发的应用程序去使用 Pro Capture 系列采集卡的扩展功能。



本文主要对软件开发工具包的函数和类型定义进行介绍，具体各项功能说明请参阅“软件开发指南”。

类型和函数索引

DirectShow 接口	类型定义请参阅第二章，接口函数说明请参阅第三章第 1 节。
MWCapture 接口	类型定义请参阅第二章，函数说明请参阅第三章第 2 节。

2 类型定义

2.1 枚举类型

2.1.1 MW_FAMILY_IDEnumerated

定义采集设备的产品系列。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCommon.h

常量定义

MW_FAMILY_ID_PRO_CAPTURE

使用 PCI-e 接口和计算机相连接的采集设备。

MW_FAMILY_ID_VALUE_CAPTURE

使用 PCI-e 接口和计算机相连接的采集设备 和 Pro Capture 系列相比 ,在软硬件上做了一定的简化 ,包括不支持多流输出、不使用板载缓存等。

MW_FAMILY_ID_USB_CAPTURE

使用 USB 接口和计算机相连接的采集设备。

2.1.2 MWCAP_PRODUCT_IDEnumerated

定义采集设备的设备类型。不同类型的设备具有不同的输入/输出接口。

原型

```
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;	

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_PRODUCT_ID_PRO_CAPTURE_AIO

具有 PCI-e 接口的输入输出采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_DVI

具有 PCI-e 主机接口和 DVI 输入接口的采集卡。 .

MWCAP_PRODUCT_ID_PRO_CAPTURE_HDMI

具有 PCI-e 主机接口和 HDMI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_SDI

具有 PCI-e 主机接口和 SDI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_SDI

具有 PCI-e 主机接口和两路 SDI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_DVI

具有 PCI-e 主机接口和两路 DVI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_DUAL_HDMI

具有 PCI-e 主机接口和两路 HDMI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_QUAD_SDI

具有 PCI-e 主机接口和四路 SDI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_QUAD_HDMI

具有 PCI-e 主机接口和四路 HDMI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_MINI_HDMI

具有 PCI-e 主机接口和迷你 HDMI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_HDMI_4K

具有 PCI-e 主机接口和 4KHDMI 输入接口的采集卡。

MWCAP_PRODUCT_ID_PRO_CAPTURE_MINI_SDI

具有 PCI-e 主机接口和迷你 SDI 输入接口的采集卡。

2.1.3 MWCAP_VIDEO_INPUT_TYPEEnumerated

定义视频通道的信号输入接口掩码。

原型

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;	

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_INPUT_TYPE_NONE

无信号输入接口。

MWCAP_VIDEO_INPUT_TYPE_HDMI

HDMI 信号输入接口。

MWCAP_VIDEO_INPUT_TYPE_VGA

VGA 信号输入接口。

MWCAP_VIDEO_INPUT_TYPE_SDI

SDI 信号输入接口。

MWCAP_VIDEO_INPUT_TYPE_COMPONENT

Component 输入信号接口。

MWCAP_VIDEO_INPUT_TYPE_CVBS

CVBS 信号输入接口。

MWCAP_VIDEO_INPUT_TYPE_YC

YC 信号输入接口。

2.1.4 MWCAP_AUDIO_INPUT_TYPEEnumerated

定义音频通道的信号输入接口掩码。

原型

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;	

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_AUDIO_INPUT_TYPE_NONE

无信号输入接口。

MWCAP_AUDIO_INPUT_TYPE_HDMI

HDMI 信号输入接口。

MWCAP_AUDIO_INPUT_TYPE_SDI

SDI 信号输入接口。

MWCAP_AUDIO_INPUT_TYPE_LINE_IN

线路输入信号输入接口。

MWCAP_AUDIO_INPUT_TYPE_MIC_IN

麦克风信号输入接口。

2.1.5 MWCAP_PCIE_LINK_TYPEEnumerated

定义采集设备的音频输入接口类型。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_PCIE_LINK_GEN_1

PCI-e 1.0 传输标准。

MWCAP_PCIE_LINK_GEN_2

PCI-e 2.0 传输标准。

MWCAP_PCIE_LINK_GEN_3

PCI-e 3.0 传输标准。

MWCAP_PCIE_LINK_GEN_4

PCI-e 4.0 传输标准。

2.1.6 MWCAP_VIDEO_TIMING_TYPEEnumerated

定义采集设备的音频输入接口类型。

原型

```
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;	

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_TIMING_NONE

视频通道无时序。

MWCAP_VIDEO_TIMING_LEGACY

视频通道使用 LEGACY 时序。

MWCAP_VIDEO_TIMING_DMT

视频通道使用 DMT 时序。

MWCAP_VIDEO_TIMING_CEA

视频通道使用 CEA 时序。

MWCAP_VIDEO_TIMING_GTF

视频通道使用 GTF 时序。

MWCAP_VIDEO_TIMING_CVT

视频通道使用 CVT 时序。

MWCAP_VIDEO_TIMING_CVT_RB

视频通道使用 CVT_RB 时序。

MWCAP_VIDEO_TIMING_FAILSAFE

视频通道使用 FAILSAFE 时序。

2.1.7 MWCAP_VIDEO_COLOR_FORMATEnumerated

定义采集设备的音频输入接口类型。

原型

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;	

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_COLOR_FORMAT_UNKNOWN

无法识别的色彩格式。

MWCAP_VIDEO_COLOR_FORMAT_RGB

RGB 色彩格式。

MWCAP_VIDEO_COLOR_FORMAT_YUV601

YUV601 色彩格式。

MWCAP_VIDEO_COLOR_FORMAT_YUV709

YUV709 色彩格式。

MWCAP_VIDEO_COLOR_FORMAT_YUV2020

YUV2020 色彩格式。

MWCAP_VIDEO_COLOR_FORMAT_YUV2020C

YUV2020C 色彩格式。

2.1.8 MWCAP_VIDEO_QUANTIZATION_RANGEEnumerated

定义视频量化范围。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_QUANTIZATION_UNKNOWN

无法识别的视频量化范围。

MWCAP_VIDEO_QUANTIZATION_FULL

全域视频量化范围。在该范围下，8 位黑白颜色范围为 0-255。

MWCAP_VIDEO_QUANTIZATION_LIMITED

局域视频量化范围。在该范围下，8 位黑白颜色范围为 16-235。

2.1.9 MWCAP_VIDEO_SATURATION_RANGEEnumerated

定义视频饱和范围。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_SATURATION_UNKNOWN

未知的视频饱和范围。

MWCAP_VIDEO_SATURATION_FULL

全域视频饱和范围。在该范围下，8 位黑白颜色范围为 0-255。

MWCAP_VIDEO_SATURATION_LIMITED

局域视频饱和范围。在该范围下，8 位黑白颜色范围为 16-235。

MWCAP_VIDEO_SATURATION_EXTENDED_GAMUT

扩展局域视频饱和范围。在该范围下，8 位黑白颜色范围为 1-254。

2.1.10 MWCAP_VIDEO_FRAME_TYPEEnumerated

定义视频帧类型。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_FRAME_2D

2 维视频帧。

MWCAP_VIDEO_FRAME_3D_TOP_AND_BOTTOM_FULL

分顶场和底场的全景 3 维视频帧。

MWCAP_VIDEO_FRAME_3D_TOP_AND_BOTTOM_HALF

分顶场和底场的半景 3 维视频帧。

MWCAP_VIDEO_FRAME_3D_SIDE_BY_SIDE_FULL

并排全景的 3 维视频帧。

MWCAP_VIDEO_FRAME_3D_SIDE_BY_SIDE_HALF

并排半景的 3 维视频帧。

2.1.11 MWCAP_VIDEO_DEINTERLACE_MODEEnumerated

定义视频信号去隔行模式。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_DEINTERLACE_WEAVE

使用 Weave 模式去隔行。

MWCAP_VIDEO_DEINTERLACE_BLEND

使用混合模式去隔行。

MWCAP_VIDEO_DEINTERLACE_TOP_FIELD

通过只取顶场数据去隔行。

MWCAP_VIDEO_DEINTERLACE_BOTTOM_FIELD

通过只取底场数据去隔行。

2.1.12 MWCAP_VIDEO_ASPECT_RATIO_CONVERT_MODEEnumerated

定义视频信号的宽高比转换模式。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_ASPECT_RATIO_IGNORE

忽略宽高比。

MWCAP_VIDEO_ASPECT_RATIO_CROPPING

通过裁剪转换宽高比。

MWCAP_VIDEO_ASPECT_RATIO_PADDING

通过填补转换宽高比。

2.1.13 MWCAP_VIDEO_SYNC_TYPEEnumerated

定义视频信号的同步类型。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

VIDEO_SYNC_ALL

完全同步。

VIDEO_SYNC_HS_VS

通过裁剪转换宽高比。

VIDEO_SYNC_CS

通过填补转换宽高比。

VIDEO_SYNC_EMBEDDED

通过填补转换宽高比。

2.1.14 MWCAP_LED_MODEEnumerated

定义 LED 模式。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_LED_AUTO

自动模式。

MWCAP_LED_OFF

LED 保持熄灭状态。

MWCAP_LED_ON

LED 保持常亮。

MWCAP_LED_BLINK

LED 保持闪烁。

MWCAP_LED_DBL_BLINK

LED 每闪烁两次停一次。

MWCAP_LED_BREATH

呼吸模式。

2.1.15 MWCAP_SD_VIDEO_STANDARDEnumerated

定义使用的电视广播制式。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_SD_VIDEO_NONE

无。

MWCAP_SD_VIDEO_NTSC_M

使用 NTSC_M 标准。

MWCAP_SD_VIDEO_NTSC_433

使用 NTSC_433 标准。

MWCAP_SD_VIDEO_PAL_M

使用 PAL_M 标准。

MWCAP_SD_VIDEO_PAL_60

使用 PAL_60 标准。

MWCAP_SD_VIDEO_PAL_COMBN

使用 PAL_COMBN 标准。

MWCAP_SD_VIDEO_PAL_BGHID

使用 PAL_BGHID 标准。

MWCAP_SD_VIDEO_SECAM

使用 SECAM 标准。

MWCAP_SD_VIDEO_SECAM_60

使用 SECAM_60 标准。

2.1.16 MWCAP_VIDEO_SIGNAL_STATEEnumerated

定义输入视频信号的状态。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_SIGNAL_STATE_NONE

无视频信号。

MWCAP_VIDEO_SIGNAL_UNSUPPORTED

无效的视频信号，采集设备检测到有输入信号，但无法对该信号锁定识别。

MWCAP_VIDEO_SINGAL_STATE_LOCKING

正在锁定视频信号，视频信号的状态有效，但还没有锁定。

MWCAP_VIDEO_SINGAL_STATE_LOCKED

已锁定的视频信号，采集设备能够采集当前输入信号。

2.1.17 MWCAP_VIDEO_FRAME_STATEEnumerated

定义视频帧的状态。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_VIDEO_FRAME_STATE_INITIAL

初始状态。

MWCAP_VIDEO_FRAME_STATE_F0_BUFFERING

正在缓冲第 0 场。

MWCAP_VIDEO_FRAME_STATE_F1_BUFFERING

正在缓冲第 1 场。

MWCAP_VIDEO_FRAME_STATE_BUFFERED

视频帧已经缓冲好。

2.1.18 MWCAP_HDMI_INFOFRAME_IDEnumerated

定义 HDMI 信号信息帧的 ID。

原型

```
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;

```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_HDMI_INFOFRAME_ID_AVI,

AVI 信息帧。

MWCAP_HDMI_INFOFRAME_ID_AUDIO

Audio 信息帧。

MWCAP_HDMI_INFOFRAME_ID_SPD

SPD 信息帧。

MWCAP_HDMI_INFOFRAME_ID_MS

MS 信息帧。

MWCAP_HDMI_INFOFRAME_ID_VS

VS 信息帧。

MWCAP_HDMI_INFOFRAME_ID_ACP

ACP 信息帧。

MWCAP_HDMI_INFOFRAME_ID_ISRC1

ISRC1 信息帧。

MWCAP_HDMI_INFOFRAME_ID_ISRC2

ISRC2 信息帧。

MWCAP_HDMI_INFOFRAME_ID_GAMUT

GAMUT 信息帧。

MWCAP_HDMI_INFOFRAME_ID_COUNT

信息帧类型总数。

2.1.19 MWCAP_HDMI_INFOFRAME_MASKEnumerated

定义 HDMI 信号信息帧的掩码。

原型

```

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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

常量定义

MWCAP_HDMI_INFOFRAME_MASK_AVI,

AVI 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_AUDIO

Audio 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_SPD

SPD 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_MS

MS 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_VS

VS 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_ACP

ACP 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_ISRC1

ISRC1 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_ISRC2

ISRC2 信息帧。

MWCAP_HDMI_INFOFRAME_MASK_GAMUT

GAMUT 信息帧。

2.2 数据结构

2.2.1 MWCAP_CHANNEL_INFO structure

该结构用于记录通道的详细信息。

原型

```
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;
```

要求

头文件 | LibMWCapture\MWCaptureExtension.h

成员

- wFamilyID**
通道所属采集设备的产品类型。
- wProductID**
通道所属采集设备的设备类型 ID。
- chHardwareVersion**
通道所属采集设备的硬件版本号。
- byFirmwareID**
通道所属采集设备的固件 ID。
- dwFirmwareVersion**
通道所属采集设备的固件版本。
- dwDriverVersion**
通道所属采集设备的驱动版本。
- szProductFamilyName**
通道所属采集设备的产品系列名称。
- szProductName**
通道所属采集设备的产品类型名称。
- szFirmwareName**

通道所属采集设备的固件名称。

szBoardSerialNo

通道所属采集设备的硬件序列号。

byBoardIndex

设备开关序号，采集设备上旋钮开关指示的序号，序号从 0 开始。

byChannelIndex

通道序号，一个采集设备上有多个通道时，序号从 0 开始。

2.2.2 MWCAP_PRO_CAPTURE_INFO structure

该结构用于记录 PCI-e 接口采集设备详细信息。

原型

```
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;
```

成员

byPCIBusID

采集设备连接到计算机上的系统总线 ID。

byPCIDevID

采集设备连接到计算机上的 PCI 设备 ID。

byLinkType

采集设备和计算机 PCI-e 接口协商一致后的数据传输标准。

byLinkWidth

采集设备连接计算机的 PCI-e 插槽 Lane 数量。

byBoardIndex

采集设备上拨码开关的数值。

wMaxPayloadSize

采集设备上的最大负载大小。

wMaxReadRequestSize

采集设备上的最大读请求大小。

cbTotalMemorySize

采集设备上的总的内存大小。

cbFreeMemorySize

采集设备上的空闲的内存大小。

2.2.3 MWCAP_VIDEO_CAPS structure

该结构用于记录视频采集能力。

原型

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

成员

dwCaps

视频采集能力。

wMaxInputWidth

输入视频的最大宽度。

wMaxInputHeight

输入视频的最大高度。

wMaxOutputWidth

输出视频的最大宽度。

wMaxOutputHeight

输出视频的最大高度。

2.2.4 MWCAP_AUDIO_CAPS structure

该结构用于记录音频采集能力。

原型

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

成员

dwCaps

视频采集能力。

2.2.5 MWCAP_FIRMWARE_STORAGE_INFO structure

该结构用于记录固件存储信息。

原型

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

成员

- cbStorage**
固件存储区域的长度。
- cbEraseBlock**
擦除区域的长度。
- cbProgramBlock**
程序块的存储区域长度。
- cbHeaderOffset**
固件头部区域的存储偏移量。

2.2.6 MWCAP_FIRMWARE_ERASE structure

该结构用于指定要擦除的固件存储区域。

原型

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

成员

- cbOffset**
要擦除的存储区域的偏移地址。
- cbErase**
擦除区域的长度。

2.2.7 MWCAP_SDI_SPECIFIC_STATUS structure

该结构用于定义 SDI 信号状态。

原型

```
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;
```

成员

sdiType

SDI 信号类型。

sdiScanningFormat

SDI 扫描格式。

sdiBitDepth

SDI 位深。

sdiSamplingStruct

SDI 采样的数据结构。

bST352DataValid

ST352 数据是否有效。

dwST352Data

ST352 数据。

2.2.8 MWCAP_HDMI_VIDEO_TIMING structure

该结构用于定义 HDMI 视频信号的时序信息。

原型

```
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;	

成员

bInterlaced

是否交错。

dwFrameDuration

帧时间间隔。

wHSyncWidth

水平方向同步宽度。

wHFrontPorch

水平方向的前沿宽度。

wHBackPorch

水平方向的后沿宽度。

wHActive

水平方向的有效宽度。

wHTotalWidth

水平方向的总宽度。

wField0VSyncWidth

第 0 场的垂直同步宽度。

wField0VFrontPorch

第 0 场的垂直方向前沿宽度。

wField0VBackPorch

第 0 场的垂直方向后沿宽度。

wField0VActive

第 0 场的垂直方向有效宽度。

wField0VTotalHeight

第 0 场的垂直方向的总高度。

wField1VSyncWidth

第 1 场的垂直同步宽度。

wField1VFrontPorch

第 1 场的垂直方向前沿宽度。

wField1VBackPorch

第 1 场的垂直方向后沿宽度。

wField1VActive

第 1 场的垂直方向有效宽度。

wField1VTotalHeight

第 1 场的垂直方向的总高度。

2.2.9 MWCAP_HDMI_SPECIFIC_STATUS structure

该结构用于定义 HDMI 信号的状态。

原型

```
typedef struct _MWCAP_HDMI_SPECIFIC_STATUS {
    BOOLEAN bHDMIMode;
    BOOLEAN bHDCP;
    BYTE byBitDepth;
    HDMI_PIXEL_ENCODING pixelEncoding;
    BYTE byVIC;
    BOOLEAN bITContent;
    BOOLEAN b3DFormat;
    BYTE by3DStructure;
    BYTE bySideBySideHalfSubSampling;
    MWCAP_HDMI_VIDEO_TIMING videoTiming;
} MWCAP_HDMI_SPECIFIC_STATUS;
```

成员

- bHDMIMode**
是否是 HDMI 模式。
- bHDCP**
是否是 HDCP 加密信号。
- byBitDepth**
位深。
- pixelEncoding**
像素格式。
- byVIC**
EDID 中的视频识别码，用于指定标准的分辨率和时序。
- bITContent**
IT Content 标志位。
- b3DFormat**
是否是 3D 格式。
- by3DStructure**
3D 结构。
- bySideBySideHalfSubSampling**
并排半采样。
- videoTiming**
视频时序。

2.2.10 MWCAP_COMPONENT_SPECIFIC_STATUS structure

该结构用于定义 Component 信号的状态。

原型

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;	

成员

syncInfo

视频同步信息。

bTriLevelSync

是否是三级同步。

videoTiming

视频时序信息。

videoTimingSettings

视频时序设置。

2.2.11 MWCAP_CVBS_YC_SPECIFIC_STATUS structure

该结构用于定义 CVBS_YC 信号的状态。

原型

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

成员

Standard

定义使用的视频标准。

b50Hz

扫描频率是否是 50Hz。

2.2.12MWCAP_INPUT_SPECIFIC_STATUS structure

该结构用于定义输入信号的状态。

原型

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;	

成员

bValid

定义该输入信号状态是否有效。

dwVideoInputType

输入视频信号的类型。

sdiStatus

SDI 信号状态。

hdmiStatus

HDMI 信号状态。

vgaComponentStatus

VGA component 信号状态。

cvbsYcStatus

CVBS-YC 信号状态。

2.2.13 MWCAP_VIDEO_SIGNAL_STATUS structure

该结构用于定义视频信号的状态。

原型

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;
```

成员

- state**
定义该视频信号的可访问情况。
- x**
水平方向的起始位置。
- y**
垂直方向的起始位置。
- cx**
视频画面的宽度。
- cy**
视频画面的高度。
- cxTotal**
总宽度。
- cyTotal**
总高度。
- bInterlaced**
是否是交错信号。
- dwFrameDuration**
视频帧的间隔时间
- nAspectX**
视频宽高比的宽度
- nAspectY**
视频宽高比的高度
- bSegmentedFrame**
是否是分段的帧
- frameType**
视频帧的类型
- colorFormat**
视频颜色格式
- quantRange**
量化范围
- satRange**
饱和度范围

2.2.14 MWCAP_AUDIO_SIGNAL_STATUS structure

该结构用于定义音频信号的状态。

原型

```
typedef struct _MWCAP_AUDIO_SIGNAL_STATUS {  
    WORD  
    wChannelValid;
```

BOOLEAN	bLPCM;
BYTE	cBitsPerSample;
DWORD	dwSampleRate;
BOOLEAN	bChannelStatusValid;
IEC60958_CHANNEL_STATUS	channelStatus;
} MWCAP_AUDIO_SIGNAL_STATUS;	

成员

wChannelValid

有效声道的 mask 值；最低位表示第 1 和第 2 声道是否有效，第二位表示 3 和 4 声道是否有效，第三位表示 5 和 6 声道是否有效，第四位表示 7 和 8 声道是否有效。

bLPCM

表示是否是 LPCM 格式。

cBitsPerSample

每个音频采样的位深。

dwSampleRate

采样率。

bChannelStatusValid

表示该通道状态是否有效。

channelStatus

该音频通道的状态。

2.2.15 MWCAP_TIMER_EXPIRE_TIME structure

该结构用于定义音频信号的状态。

原型

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

成员

pvTimer

计时器句柄。

llExpireTime

预设的超时时间。

2.2.16MWCAP_TIMER_EXPIRE_TIME structure

该结构用于注册事件。

原型

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

成员

Property

返回 KSPROPERTY 对象。

hEvent

注册的事件的句柄。

ullEnableBits

事件类型的 mask 值。参考 MWCAP_NOTIFY_INPUT_SOURCE_START_SCAN 等

2.2.17 MWCAP_SMPTE_TIMECODE structure

该结构用于定义 SMPTE 时间码。

原型

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

成员

byFrames

帧号。

bySeconds

秒。

byMinutes

分钟

byHours

小时

2.2.18 MWCAP_VIDEO_BUFFER_INFO structure

该结构用于定义视频缓冲信息。

原型

```
typedef struct _MWCAP_VIDEO_BUFFER_INFO {
```

DWORD	cMaxFrames;
BYTE	iNewestBuffering;
BYTE	iBufferingFieldIndex;
BYTE	iNewestBuffered;
BYTE	iBufferedFieldIndex;
BYTE	iNewestBufferedFullFrame;
DWORD	cBufferedFullFrames;
} MWCAP_VIDEO_BUFFER_INFO;	

成员

cMaxFrames

板载缓存中的最大帧数。

iNewestBuffering

正在缓存的片序号。一帧视频数据可能包含多个片。

iBufferingFieldIndex

正在缓存的场的序号

iNewestBuffered

最新缓冲好的片序号

iBufferedFieldIndex

最新缓冲好的场序号

iNewestBufferedFullFrame

最新缓冲好的帧序号

cBufferedFullFrames

缓冲好的整帧数量

2.2.19 MWCAP_VIDEO_FRAME_INFO structure

该结构用于定义视频帧信息。

原型

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;
```

成员

state

表示该视频帧的状态。

bInterlaced

是否是隔行的。

bSegmentedFrame

是否是分段的帧。

bTopFieldFirst

是否是顶场在前。

bTopFieldInverted

顶场是否反转。

cx

视频帧的宽度

cy

视频帧的高度

nAspectX

宽高比中的宽度

nAspectY

宽高比中的高度

allFieldStartTimes

顶场和底场分别开始采集的时间

allFieldBufferedTimes

顶场和底场分别缓冲好的时间

aSMPTETimeCodes

顶场和底场各自的时间码

2.2.20 MWCAP_VIDEO_CAPTURE_OPEN structure

该结构用于定义开始采集的事件参数。

原型

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

成员

hEvent

时间句柄。

2.2.21 MWCAP_VIDEO_CAPTURE_FRAME structure

该结构用于定义视频帧的采集参数。

原型

```
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;
```

成员

dwFOURCC

色彩格式。参考 MWFOURCC.h。

cx

采集宽度。

cy
采集高度。

nAspectX
宽高比中的宽度

nAspectY
宽高比中的高度

colorFormat
颜色格式标准。

quantRange
量化范围。

satRange
饱和度范围。

sContrast
对比度。

sBrightness
亮度。

sSaturation
饱和度。

sHue
色度。

rectSource
采集的源区域。

rectTarget
采集的目标区域。

deinterlaceMode
去隔行模式。

aspectRatioConvertMode
宽高比转换模式。

iSrcFrame
采集的帧序号。

pOSDImage
用于 OSD 合成的图片的句柄，通过 MWCAP_KSPROPERTY_VIDEO_CREATE_IMAGE 接口获得。

aOSDRects
OSD 图片的合成的目标区域。

cOSDRects
OSD 图片的合成的目标区域个数。

bPhysicalAddress
是否将采集到的数据存放在指定的物理地址中。

pvFrame
采集到的帧数据被存放的内存地址。

liPhysicalAddress
采集到的帧数据被存放的物理地址。

cbFrame
这一帧数据的长度。

cbStride

这一帧数据所使用的步长。

bBottomUp

是否上下颠倒。

cyPartialNotify

部分通知的个数。

dwProcessSwitchs

视频处理的掩码值。参考 MWCAP_VIDEO_PROCESS_xx。

pvContext

采集该帧视频的上下文环境。

2.2.22 MWCAP_VIDEO_CAPTURE_STATUS structure

该结构用于定义视频采集的状态。

原型

```
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;
```

成员

pvContext

视频采集的上下文环境。

bPhysicalAddress

是否使用物理地址作存放采集到的数据。

pvFrame

存放采集数据的内存地址。

liPhysicalAddress

存放采集数据的物理地址。

iFrame

正在采集的帧的序号。

bFrameCompleted

一帧是否采集完成。

cyCompleted

采集完成的帧数量。

cyCompletedPrev

先前采集完成的帧数量。

2.2.23 MWCAP_AUDIO_CAPTURE_FRAME structure

该结构用于定义音频采集的帧信息。

原型

```
typedef struct _MWCAP_AUDIO_CAPTURE_FRAME {  
    DWORD                cFrameCount;  
    DWORD                iFrame;  
    DWORD                dwSyncCode;  
    DWORD                dwReserved;  
    LONGLONG             llTimestamp;  
    DWORD                adwSamples[MWCAP_AUDIO_SAMPLES_PER_FRAME*  
                                   MWCAP_AUDIO_MAX_NUM_CHANNELS];  
} MWCAP_AUDIO_CAPTURE_FRAME;
```

成员

cFrameCount

缓存的帧数量。

iFrame

当前这一帧的索引号。

dwSyncCode

音频帧数据的同步码。

dwReserved

保留字段。

llTimestamp

该音频帧的时间戳。

adwSamples

音频采样数据。每个采样 32 比特宽，高位有效。通道顺序为，左 0，左 1，左 2，左 3，右 0，右 1，右 2，右 3。

2.2.24 MWCAP_VIDEO_ASPECT_RATIO structure

该结构用于定义视频宽高比。

原型

```
typedef struct _MWCAP_VIDEO_ASPECT_RATIO {  
    int                nAspectX;  
    int                nAspectY;
```

```
} MWCAP_VIDEO_ASPECT_RATIO;
```

成员

nAspectX

宽度比。

nAspectY

高度比。

2.2.25 MWCAP_VIDEO_CONNECTION_FORMAT structure

该结构用于定义视频连接格式。

原型

```
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;
```

成员

bConnected

表示是否连接。

cx

视频画面的宽度。

cy

视频画面的高度。

dwFrameDuration

视频帧的间隔时间

dwFOURCC

色彩空间。参考 MWFOURCC.h。

nAspectX

视频宽高比的宽度

nAspectY

视频宽高比的高度

colorFormat

视频颜色格式标准

quantRange

量化范围

satRange

饱和度范围

2.2.26 MWCAP_VIDEO_PROCESS_SETTINGS structure

该结构用于定义视频处理设置。

原型

```
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;
```

成员

dwProcessSwitchs

视频处理的掩码值。参考 MWCAP_VIDEO_PROCESS_xx。

rectSource

要处理的源区域。

nAspectX

视频宽高比的宽度

nAspectY

视频宽高比的高度

bLowLatency

是否启用低延迟模式

colorFormat

视频颜色格式标准

quantRange

量化范围

satRange

饱和度范围

deinterlaceMode

去隔行模式。

aspectRatioConvertMode

宽高比转换模式。

2.2.27 MWCAP_KSPROPERTY_VIDEO_CREATE_IMAGE_S structure

该结构用于定义在板载内存中创建画面的参数。

原型

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

成员

Property

KSPROPERTY 对象。

cx

画面的宽度。

cy

画面的高度。

2.2.28 MWCAP_VIDEO_UPLOAD_IMAGE structure

该结构用于定义向采集卡上载图像的参数。

原型

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;	

成员

pvDestImage

目标图像句柄。

cfDest

色彩格式标准。

xDest

目标区域的水平方向起始位置。

yDest

目标区域的垂直方向起始位置。

cxDest

目标区域的宽度。

cyDest

目标区域的高度。

quantRangeDest

目标图像的量化范围。

satRangeDest

目标图像的饱和度范围

bSrcPhysicalAddress

原视频是否使用的物理地址。

pvSrcFrame

原视频数据的内存地址。

liSrcPhysicalAddress

原视频数据的物理地址。

cbSrcFrame

原视频数据的总长度。

cbSrcStride

原视频数据的步长。

cxSrc

原视频的宽度。

cySrc

原视频的高度。

bSrcBottomUp

原视频是否上下颠倒。

bSrcPixelAlpha

原视频像素是否具有 Alpha 分量。

bSrcPixelXBGR

原视频像素是否是 XBGR 格式。

2.2.29 MWCAP_VIDEO_OSD_SETTINGS structure

该结构用于定义 OSD 设置。

原型

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

成员

bEnable

是否启用 OSD。

wszPNGFilePath

加载 PNG 图片的文件路径。

2.2.30 MWCAP_VIDEO_OSD_IMAGE structure

该结构用于定义 OSD 图片信息。

原型

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

成员

pvOSDImage

OSD 图片的句柄。

aOSDRects

需要合成该图片的目标区域。

cOSDRects

需要合成该图片的目标区域数量。

2.2.31 MWCAP_VIDEO_CUSTOM_TIMING structure

该结构用于定义自定义视频时序。

原型

```
typedef struct _MWCAP_VIDEO_CUSTOM_TIMING {  
    MWCAP_VIDEO_SYNC_INFO        syncInfo;
```

MWCAP_VIDEO_TIMING_SETTINGS	videoTimingSettings;
} MWCAP_VIDEO_CUSTOM_TIMING;	

成员

- syncInfo**
视频同步信息。
- videoTimingSettings**
视频时序的设置信息。

2.2.32 MWCAP_VIDEO_SYNC_INFOstructure

该结构用于记录视频时序的同步信息。

原型

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

成员

- bySyncType**
视频时序的同步类型。
- bHSPolarity**
视频时序的水平方向同步极性。
- bVSPolarity**
视频时序的竖直方向同步极性。
- bInterlaced**
视频时序是否交错。
- dwFrameDuration**
视频时序的帧间隔时间。
- wVSyncLineCount**
视频时序的竖直方向同步扫描线数量。
- wFrameLineCount**
视频时序的帧扫描线数量。

2.2.33 MWCAP_VIDEO_TIMING structure

该结构用于记录视频通道的时序信息。

原型

```
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;
```

成员

dwType

视频通道时序类型。

dwPixelClock

视频通道时序的像素时钟。

bInterlaced

视频通道时序是否交错。

bySyncType

视频通道时序的同步类型。

bHSPolarity

视频通道时序的水平方向同步极性是否为正。

bVSPolarity

视频通道时序的垂直方向同步极性是否为正。

wHActive

视频通道时序的水平方向有效时间。

wHFrontPorch

视频通道时序的水平方向前沿。

wHSyncWidth

视频通道时序的水平方向同步宽度。

wHBackPorch

视频通道时序的水平方向后沿。

wVActive

视频通道时序的垂直方向有效时间。

wVFrontPorch

视频通道时序的竖直方向前沿。

wVSyncWidth

视频通道时序的竖直方向同步宽度。

wVBackProch

视频通道时序的水平方向后沿。

2.2.34 MWCAP_VIDEO_TIMING_SETTINGS structure

该结构用于记录视频时序的设置信息。

原型

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;	

成员

wAspectX

视频宽高比中的宽度。

wAspectY

视频宽高比中的高度。

x

水平方向的起始位置。

y

竖直方向的起始位置。

cx

宽度。

cy

高度。

cxTotal

水平方向总的宽度。

byClampPos

钳制位置。

2.2.35 MWCAP_KSPROPERTY_DWORD_S structure

该结构用于在指定 KSPROPERTY 的同时附带一个 DWORD 参数。

原型

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

成员

Property

KSPROPERTY 对象，用于调用接口。

dwParameter

附加的 DWORD 参数。

2.2.36 MWCAP_KSPROPERTY_PTR64_S structure

该结构用于在指定 KSPROPERTY 的同时附带一个 64 位指针参数。

原型

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

成员

Property

KSPROPERTY 对象，用于调用接口。

pvParameter

附加的 64 位指针参数。

2.2.37 MWCAP_DWORD_PARAMETER_RANGE structure

该结构用于定义参数的范围,步长和默认值。

原型

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

成员

dwMin

最小值。

dwMax

最大值。

dwStep

步长。

dwDefault

默认值。

2.2.38 MWCAP_DWORD_PARAMETER_VALUE structure

该结构用于定义一种 DWORD 参数值。

原型

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

成员

dwFlags

标识。

dwValue

值。

3 函数说明

3.1 Direct Show 扩展接口函数

3.1.1 GetChannelInfo

该函数用于获取采集通道的信息

函数原形

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

参数

pChannelInfo [out]
返回采集设备上的通道信息。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_CHANNEL_INFO 结构的描述。

3.1.2 GetFamilyInfo

该函数用于获取采集通道的产品系列信息

函数原形

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

参数

pvFamilyInfo [out]
返回采集通道的产品系列信息，对于 Pro Capture 系列采集卡传入 MWCAP_PRO_CAPTURE_INFO 结构地址指针。

pcbFamilyInfo [in]
pvFamilyInfo 内存字节长度。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.3 GetVideoCaps

该函数用于获取视频通道的性能属性

函数原形

```
BOOL GetVideoCaps(  
    MWCAP_VIDEO_CAPS *    pVideoCaps  
);
```

参数

pVideoCaps [out]
返回视频通道的性能属性。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_CAPS 结构的描述。

3.1.4 GetAudioCaps

该函数用于获取音频通道的性能属性

函数原形

```
BOOL GetAudioCaps (  
    MWCAP_AUDIO_CAPS *    pAudioCaps  
);
```

参数

pAudioCaps [out]
返回音频通道的性能属性。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_AUDIO_CAPS 结构的描述。

3.1.5 GetDeviceInstanceID

该函数用于获取通道对应的系统设备 ID

函数原形

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

参数

pszInstanceID [out]
 返回通道对应的系统设备 ID。
pcbInstanceID [in]
 pszInstanceID 变量内存字节长度。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.6 RegisterNotify

该函数用于注册异步事件通知对象

函数原形

```
BOOL RegisterNotify(  
    HANDLE      hEvent,  
    ULONGLONG   ullEnableBits,  
    MWCAP_PTR64& pvNotifyEvent  
);
```

参数

- hEvent [in]
事件对象。
- ullEnableBits [in]
事件响应类型值。
- pvNotifyEvent [out]
异步事件通知对象。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.7 UnregisterNotify

该函数用于注销异步事件通知对象

函数原形

```
BOOL UnregisterNotify(  
    MWCAP_PTR64    pvNotifyEvent  
);
```

参数

- pvNotifyEvent [in]
异步事件通知对象。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.8 GetNotifyStatus

该函数用于获取事件通知的状态

函数原形

```
BOOL GetNotifyStatus(  

```


MWCAP_PTR64 ULONGLONG&);	pvNotifyEvent, ullStatusBits
---------------------------------	---------------------------------

参数

pvNotifyEvent [in]

采集设备上的通道信息。

ullStatusBits [out]

事件响应类型值。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.9 GetVideoInputSourceCount

该函数用于获取视频通道的输入接口数目

函数原形

BOOL GetVideoInputSourceCount(DWORD * pcInputSources);

参数

pcInputSources [out]

输入接口数目。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.10 GetVideoInputSourceArray

该函数用于获取视频通道的输入接口

函数原形

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

参数

pdwInputSources [out]

视频输入源索引值。

pcBufferItems [out]

视频输入源类型个数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.11 GetAudioInputSourceCount

该函数用于获取音频通道的输入接口数目

函数原形

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

参数

pcInputSources [in]

输入接口数目。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_CHANNEL_INFO 结构的描述。

3.1.12 GetAudioInputSourceArray

该函数用于获取音频通道的输入接口

函数原形

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

参数

pdwInputSources [out]

音频输入源索引值。

pcBufferItems [out]

音频输入源类型个数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.13 SetInputSourceScan

该函数用于设置采集通道的输入接口扫描类型

函数原形

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

参数

bInputSourceScan [in]

设置是否启用自动扫描视频信号。

返回值

返回 HRESULT 值，可能的返回值如下表所示.

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.14 GetInputSourceScan

该函数用于获取采集通道的输入接口扫描类型

函数原形

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

参数

pbInputSourceScan [out]
 返回是否启用自动扫描视频信号。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.15 GetInputSourceScanState

该函数用于获取采集通道的输入接口扫描状态

函数原形

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

参数

pbInputSourceScanning [out]
 返回扫描状态。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.16 SetAVInputSourceLink

该函数用于设置当前音频通道是否和当前视频通道一致

函数原形

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

参数

bAVLink [in]

当前音视频通道是否保持一致。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.17 GetAVInputSourceLink

该函数用于获取当前音频通道是否和当前视频通道一致

函数原形

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

参数

pbAVLink [out]

当前音视频通道是否保持一致。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.18 SetVideoInputSource

该函数用于设置视频通道当前使用的输入接口

函数原形

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

参数

dwInputSource [in]
视频输入接口。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.19 GetVideoInputSource

该函数用于获取视频通道当前使用的输入接口

函数原形

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

参数

pdwInputSource [out]
视频输入接口。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.20 SetAudioInputSource

该函数用于设置音频通道当前使用的输入接口

函数原形

```
BOOL SetAudioInputSource(  
    DWORD      dwInputSource
```

```
);
```

参数

dwInputSource [in]
音频输入接口。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.21 GetAudioInputSource

该函数用于获取音频通道当前使用的输入接口

函数原形

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

参数

pdwInputSource [out]
音频输入接口。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.22 SetEDID

设置 HDMI 接口的 EDID 数据。

函数原形

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

```
);
```

参数

pbyEDID [in]
EDID 二进制数据。

cbEDID [in]
pbyEDID 数据内存字节大小。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.23 GetEDID

该函数用于获取 HDMI 接口的 EDID 数据。

函数原形

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

参数

pbyEDID [out]
EDID 二进制数据。

cbEDID [out]
pbyEDID 数据内存字节大小。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.24 GetInputSpecificStatus

该函数用于获取输入信号的状态

函数原形

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

参数

pStatus [out]
 返回指定输入信号状态。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_INPUT_SPECIFIC_STATUS 结构的描述。

3.1.25 GetVideoSignalStatus

该函数用于获取视频输入信号的状态

函数原形

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

参数

pStatus [out]
 返回视频输入信号状态。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_SIGNAL_STATUS 结构的描述。

3.1.26 GetAudioSignalStatus

该函数用于获取音频输入信号的状态

函数原形

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

参数

pStatus [out]
 返回音频输入信号状态。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_AUDIO_SIGNAL_STATUS 结构的描述。

3.1.27 GetHDMIInfoFrameValidFlags

该函数用于获取 HDMI InfoFrame 有效状态

函数原形

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

参数

pdwValidFlags [out]
 HDMI InfoFrame 有效状态。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.28 GetHDMIInfoFramePacket

该函数用于获取 HDMI InfoFrame 数据

函数原形

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

参数

id [in]
 参考MWCAP_HDMI_INFOFRAME_ID。
pInfoFramePacket [out]
 HDMI InfoFrame 包信息。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 HDMI_INFOFRAME_PACKET 结构的描述。

3.1.29 SetVideoInputAspectRatio

该函数用于设置视频输入信号的宽高比

函数原形

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

参数

pAspectRatio [in]
 视频的宽高比。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_ASPECT_RATIO 结构的描述。

3.1.30 GetVideoInputAspectRatio

该函数用于获取视频输入信号的宽高比

函数原形

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

参数

pAspectRatio [out]
视频的宽高比。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_ASPECT_RATIO 结构的描述。

3.1.31 SetVideoInputColorFormat

该函数用于设置视频输入信号的色彩空间格式

函数原形

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

参数

colorFormat [in]
视频色彩空间格式。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_COLOR_FORMAT 枚举的描述。

3.1.32 GetVideoInputColorFormat

该函数用于获取视频输入信号的色彩空间格式

函数原形

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

参数

pColorFormat [out]
 视频色彩空间格式。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_COLOR_FORMAT 枚举的描述。

3.1.33 SetVideoInputQuantizationRange

该函数用于设置视频输入信号的量化范围

函数原形

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

参数

quantRange [in]
 输入视频的量化范围。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_QUANTIZATION_RANGE 枚举的描述。

3.1.34 GetVideoInputQuantizationRange

该函数用于获取视频输入信号的量化范围

函数原形

```
BOOL GetVideoInputQuantizationRange(  
    MWCAP_VIDEO_QUANTIZATION_RANGE *    pQuantRange  
);
```

参数

pQuantRange [out]
 输入视频的量化范围。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_QUANTIZATION_RANGE 枚举的描述。

3.1.35 GetVideoCaptureConnectionFormat

获取视频采集格式信息

函数原形

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

参数

pFormat [out]
 视频采集格式。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_CONNECTION_FORMAT 枚举的描述。

3.1.36 SetVideoCaptureProcessPreset

该函数用于设置视频采集处理预设参数

函数原形

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

参数

pSettings [in]
 预设视频采集处理参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.37 GetVideoCaptureProcessPreset

该函数用于获取视频采集处理预设参数

函数原形

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

参数

pSettings [out]
 预设视频采集处理参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.38 SetVideoCaptureProcessSettings

该函数用于设置视频采集处理配置参数

函数原形

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

参数

pSettings [in]
 视频采集处理参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.39 GetVideoCaptureProcessSettings

该函数用于获取视频采集处理配置参数

函数原形

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

参数

pSettings [out]
 视频采集处理参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.40 SetVideoCaptureOSDPreset

该函数用于设置视频采集 OSD 预设参数

函数原形

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

参数

pSettings [in]
 预设视频采集 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.41 GetVideoCaptureOSDPreset

该函数用于获取视频采集 OSD 预设参数

函数原形

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

参数

pSettings [out]
 预设视频采集 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.42 SetVideoCaptureOSDSettings

该函数用于设置视频采集 OSD 配置参数

函数原形

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

参数

pSettings [in]
 视频采集 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.43 GetVideoCaptureOSDSettings

该函数用于获取视频采集 OSD 配置参数

函数原形

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

参数

pSettings [out]
 视频采集 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

Return HRESULT value. Possible values include the following.

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.44 GetVideoPreviewConnectionFormat

该函数用于获取视频预览格式信息

函数原形

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

参数

pFormat [out]
 视频预览格式。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_CONNECTION_FORMAT 结构的描述。

3.1.45 SetVideoPreviewProcessPreset

设置视频预览处理预设参数

函数原形

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

参数

pSettings [in]
 预设视频预览参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.46 GetVideoPreviewProcessPreset

该函数用于获取视频预览处理预设参数

函数原形

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

参数

pSettings [out]
 预设视频预览参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.47 SetVideoPreviewProcessSettings

该函数用于设置视频预览处理配置参数

函数原形

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

参数

pSettings [in]
 视频预览处理参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.48 GetVideoPreviewProcessSettings

该函数用于获取视频预览处理配置参数

函数原形

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

参数

pSettings [out]
 视频预览处理参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_PROCESS_SETTINGS 结构的描述。

3.1.49 SetVideoPreviewOSDPreset

该函数用于设置视频预览 OSD 预设参数

函数原形

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

参数

pSettings [in]
 视频预览 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.50 GetVideoPreviewOSDPreset

该函数用于获取视频预览 OSD 预设参数

函数原形

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

参数

pSettings [out]
 视频预览 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.51 SetVideoPreviewOSDSettings

该函数用于设置视频预览 OSD 配置参数

函数原形

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

参数

pSettings [in]
 视频预览 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.52 GetVideoPreviewOSDSettings

该函数用于获取视频采集 OSD 配置参数

函数原形

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

参数

pSettings [out]
 视频预览 OSD 参数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_OSD_SETTINGS 结构的描述。

3.1.53 SetVideoAutoHAlign

该函数用于设置视频水平方向是否自动调整

函数原形

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

参数

bAutoHAlign [in]
 视频水平方向是否自动调整。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.54 GetVideoAutoHAlign

该函数用于获取视频水平方向是否自动调整

函数原形

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

参数

pbAutoHAlign [out]
视频水平方向是否自动调整。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.55 SetVideoSamplingPhase

该函数用于设置视频采样相位大小

函数原形

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

参数

bySamplingPhase [in]
视频采样相位值。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.56 GetVideoSamplingPhase

该函数用于获取视频采样相位大小

函数原形

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

参数

pbySamplingPhase [out]
视频采样相位值。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.57 SetVideoSamplingPhaseAutoAdjust

该函数用于设置视频采样相位是否自动调整

函数原形

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

参数

bAuto [in]
是否自动调整视频采样相位。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.58 GetVideoSamplingPhaseAutoAdjust

该函数用于获取视频采样相位是否自动调整

函数原形

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

参数

pbAuto [out]
是否自动调整视频采样相位。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.59 SetVideoTiming

该函数用于设置视频时钟参数

函数原形

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

参数

pTiming [in]
视频时序信息。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_TIMING 结构的描述。

3.1.60 GetPreferredVideoTimings

该函数用于获取预设置视频时序参数

函数原形

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

参数

pTiming [out]

返回预设的视频时序。

pcTimings [out]

返回预设的视频时序个数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_TIMING 结构的描述。

3.1.61 SetCustomVideoTiming

该函数用于设置自定义视频时序参数

函数原形

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

参数

pTiming [in]

设置当前自定义视频时序。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
------	-------

S_FALSE

调用失败。

备注

具体参考 MWCAP_VIDEO_CUSTOM_TIMING 结构的描述。

3.1.62 GetCustomVideoTimingsCount

该函数用于获取自定义视频时序个数

函数原形

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

参数

pcTimings [out]
自定义视频时序个数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.63 GetCustomVideoTimingsArray

该函数用于获取自定义视频时序信息

函数原形

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

参数

pTiming [out]
包含自定义视频时序信息的数组。
pcTimings [out]
返回自定义视频时序个数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_CUSTOM_TIMING 结构的描述。

3.1.64 SetCustomVideoTimingsArray

该函数用于设置自定义视频时序

函数原形

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

参数

- pTiming [in]
 自定义视频时序。
- cTimings [in]
 自定义视频时序个数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

具体参考 MWCAP_VIDEO_CUSTOM_TIMING 结构的描述。

3.1.65 GetCustomVideoResolutionsCount

该函数用于获取自定义的视频分辨率的种数

函数原形

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

参数

- pcResolutions [out]

自定义的视频分辨率的种数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.66 GetCustomVideoResolutionsArray

该函数用于获取自定义视频分辨率

函数原形

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

参数

pResolutions [out]

包含自定义视频分辨率信息的数组。

pcResolutions [out]

返回自定义视频分辨率种数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.67 SetCustomVideoResolutionsArray

该函数用于设置自定义视频分辨率

函数原形

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

);

参数

pResolutions [in]
包含自定义视频分辨率信息的数组。

cResolutions [in]
自定义视频分辨率的种数。

返回值

返回 HRESULT 值，可能的返回值如下表所示

S_OK	调用成功。
S_FALSE	调用失败。

备注

无

3.1.68 GetCoreTemperature

该函数用于获取板载卡的当前温度

函数原形

voidGetCoreTemperature (
 LONG * pTemperatureDegC_X10
);

参数

pTemperatureDegC_X10[out]
返回温度值。

返回值

无

备注

无

3.2MWCapture 接口函数

3.2.1 MWGetVersion method

该函数用于获取 MWCapture 接口的版本号。

函数原形

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

参数

pbyMaj [out]

主版本号。

pbyMin [out]

次版本号。

pwBuild [out]

编译版本号。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

该函数始终返回 MW_SUCCEEDED. 如果参数不合法则不填入值

3.2.2 MWCaptureInitInstancemethod

初始化 MWCapture 接口。

函数原形

```
BOOL
LIBMWCAPTURE_API
MWCaptureInitInstance(
    );
```

参数

无。

返回值

返回 BOOL 值，可能的返回值如下表所示

TRUE	成功。
FALSE	失败。

备注

无

3.2.3 MWCaptureExitInstancemethod

退出 MWCapture，释放所有的资源。

函数原形

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

参数

无。

返回值

无。

备注

无

3.2.4 MWRefreshDevicemethod

该函数用于重新枚举计算机上的采集卡。

函数原形

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

参数

无。 .

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.5 MWGetChannelCountmethod

该函数用于获取总的通道数。

函数原形

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

参数

无。

返回值

返回通道数。

备注

无。

3.2.6 MWGetChannelInfoByIndexmethod

该函数用于按序号获取通道信息。

函数原形

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

参数

nIndex [in]

通道的序号，从 0 到 MWGetChannelCount() - 1。

pChannelInfo [out]

返回通道信息。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.7 MWGetFamilyInfoByIndexmethod

该函数用于按序号获取采集设备所属系列的信息。

函数原形

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

参数

- nIndex [in]
 通道的序号，从 0 到 MWGetChannelCount() - 1。
- pFamilyInfo [out]
 指向 MWCAP_PRO_CAPTURE_INFO 结构体的指针，返回系列信息。
- dwSize [in]
 结构体 MWCAP_PRO_CAPTURE_INFO 的大小。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.8 MWGetDevicePathmethod

该函数用于按序号获取采集设备的实例路径。

函数原形

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

参数

nIndex [in]
通道的序号，从 0 到 MWGetChannelCount() - 1。

pDevicePath [out]
返回设备实例路径。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.9 MWOpenChannelmethod

该函数用于按拨码开关序号和通道序号打开采集通道。

函数原形

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

参数

byBoardIndex [in]
采集设备上的拨码开关序号，该数值从 0 到 16。

byChannelIndex [in]
通道的序号，该数值从 0 开始。

返回值

调用成功，则返回视频通道句柄，调用失败，返回空值。

备注

如果在一台计算机上有两个采集设备的拨码开关序号一样，则会打开 PCI-e 插槽顺序靠前的采集设备。
同一个通道可以同时被多次打开，这些打开同一通道的句柄相互独立，互不影响。

3.2.10 MWOpenChannelByPathmethod

该函数用于按设备实例路径打开采集通道。

函数原形

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

参数

pszDevicePath [in]
设备实例路径。

返回值

调用成功，则返回视频通道句柄，调用失败，返回空值。

备注

无。

3.2.11 MWCloseChannelmethod

该函数用于关闭通道。

函数原形

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

参数

hChannel [in]
已经被打开的通道句柄。

返回值

无。

备注

无。

3.2.12 MWGetChannelInfomethod

该函数用于获取通道的信息。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

pChannelInfo [out]
 返回通道信息。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.13 MWGetFamilyInfomethod

该函数用于获取采集卡系列的信息。

函数原形

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

参数

hChannel [in]

已经被打开的通道句柄。

pFamilyInfo [out]

返回通道所属的采集卡的系列信息。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.14 MWGetVideoCapsmethod

该函数用于获取该通道的视频采集能力。

函数原形

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

参数

hChannel [in]

已经被打开的通道句柄。

pVideoCaps [out]

返回视频采集能力。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.15 MWGetAudioCapsmethod

该函数用于获取该通道的音频采集能力。

函数原形

```
MW_RESULT
LIBMWCAPTURE_API
MWGetAudioCaps(
    HCHANNEL                hChannel,
    MWCAP_AUDIO_CAPS*       pAudioCaps
);
```

参数

hChannel [in]
 已经被打开的通道句柄。

pAudioCaps [out]
 返回音频采集能力。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.16 MWGetVideoInputSourceArraymethod

该函数用于获取该通道的视频输入信号源信息。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

pdwInputSource [out]
 返回视频输入信号源。

pdwInputCount[out]
 返回视频输入信号源数量。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.17 MWGetAudioInputSourceArraymethod

该函数用于获取该通道的音频输入信号源信息。

函数原形

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

参数

- hChannel [in]
 已经被打开的通道句柄。
- pdwInputSource [out]
 返回音频输入信号源。
- pdwInputCount[out]
 返回音频输入信号源数量。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.18 MWGetInputSourceScanmethod

该函数用于获取扫描输入源的状态。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

pbScan [out]
 返回是否正在扫描。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.19 MWSetInputSourceScanmethod

该函数用于设置扫描输入源的状态。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

bScan [in]
 返回是否正在扫描。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.20 MWGetAVInputSourceLinkmethod

该函数用于获取输入源的连接状态。

函数原形

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

参数

hChannel [in]

已经被打开的通道句柄。

pbLink [out]

返回是否连接。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.21 MWSetAVInputSourceLinkmethod

该函数用于设置输入源的连接状态。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWSetAVInputSourceLink(
HCHANNEL	hChannel,
BOOLEAN	bLink

```
);
```

参数

hChannel [in]
 已经被打开的通道句柄。

bLink [in]
 指定是否连接。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.22 MWGetVideoInputSourceMethod

该函数用于获取该通道当前的视频输入源。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

pdwSource [out]
 返回当前的视频输入源。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.23 MWSetVideoInputSourceMethod

该函数用于设置该通道当前的视频输入源。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWSetVideoInputSource(
HCHANNEL	hChannel,
DWORD	dwSource
);	

参数

hChannel [in]
 已经被打开的通道句柄。

dwSource [in]
 设置当前的视频输入源。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.24 MWGetAudioInputSourceMethod

该函数用于获取该通道当前的音频输入源。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

pdwSource [out]
 返回当前的音频输入源。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.25 MWSetAudioInputSourcemethod

该函数用于设置该通道当前的音频输入源。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWSetAudioInputSource(
HCHANNEL	hChannel,
DWORD	dwSource
);	

参数

- hChannel [in]
 已经被打开的通道句柄。
- dwSource [in]
 设置当前的音频输入源。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.26 MWGetEDIDmethod

该函数用于获取该通道的 EDID 数据。

函数原形

MW_RESULT
LIBMWCAPTURE_API

MWGetEDID(HCHANNEL BYTE* ULONG*);	hChannel, pbyData, pulSize
---	----------------------------------

参数

hChannel [in]
已经被打开的通道句柄。

pbyData [out]
返回 EDID 数据。

pulSize [out]
返回 EDID 数据长度。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.27 MWSetEDIDmethod

该函数用于设置该通道的 EDID 数据。

函数原形

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

参数

hChannel [in]
已经被打开的通道句柄。

pbyData [in]
新的 EDID 数据。

ulSize [in]
EDID 数据长度。


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

参数

hChannel [in]
 已经被打开的通道句柄。

pSignalStatus [out]
 返回视频信号状态。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.30 MWGetAudioSignalStatusmethod

该函数用于获取该通道的音频信号状态。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

pSignalStatus [out]
 返回音频信号状态。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.31 MWGetHDMIInfoFrameValidFlagmethod

该函数用于获取 HDMI 信息帧的有效标志位。

函数原形

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

参数

hChannel [in]
 已经被打开的通道句柄。

pdwValidFlag [out]
 返回 HDMI 信息帧的有效标志位。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.32 MWGetHDMIInfoFramePacketmethod

该函数用于获取 HDMI 信息帧的数据包。

函数原形

```
MW_RESULT
LIBMWCAPTURE_API
MWGetHDMIInfoFramePacket(
    HCHANNEL                hChannel,
    MWCAP_HDMI_INFOFRAME_ID id,
    HDMI_INFOFRAME_PACKET*  pPacket
);
```

```
);
```

参数

hChannel [in]
 已经被打开的通道句柄。

id [in]
 HDMI 信息帧的 ID。

pPacket [out]
 返回 HDMI 信息帧的数据包。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.33 MWGetDeviceTimemethod

该函数用于返回采集设备时钟的时间值。

函数原形

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

参数

hChannel [in]
 已经被打开的视频通道句柄。

pllTime[out]
 返回采集设备时钟的时间值，它以 100 纳秒为单位。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

同一个采集设备上的不同通道使用统一的时钟，它对音视频数据同步和多通道数据同步具有重要作用。

3.2.34 MWSetDeviceTimemethod

该函数用于设置采集设备的时钟值。

函数原形

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

参数

- hChannel [in]
 已经被打开的视频通道句柄。
- lTime[in]
 设置的采集设备时钟的时间值，它以 100 纳秒为单位。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

该函数主要用于同步不同采集设备上的时钟。

3.2.35 MWRegulateDeviceTimemethod

该函数用于调节采集设备的时钟值。

函数原形

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

参数

hChannel [in]
已经被打开的视频通道句柄。

llTime[in]
调节的采集设备时钟的时间值，它以 100 纳秒为单位。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

该函数主要用于同步不同采集设备上的时钟。

该函数和 MWSetDeviceTime 的区别是，该函数设置后，采集设备的时钟值不会立即变为设置的时间值，而是以渐进方式逐步调节；而 MWSetDeviceTime 设置后，采集设备的时钟值会立即变为设置的时间值。

3.2.36 MWRegisterTimer

该函数用于注册一个时间事件对象。

函数原形

HTIMER	
LIBMWCAPTURE_API	
MWRegisterTimer(
HCHANNEL	hChannel,
HANDLE	hEvent
);	

参数

hChannel [in]
已经被打开的视频通道句柄。

hEvent [in]
一个事件句柄，它需要使用 Win32 的函数 CreateEvent()进行创建。

返回值

调用成功，返回时间事件对象的句柄，否则返回 NULL。

备注

无。

3.2.37 MWUnregisterTimermethod

该函数用于注销一个时间事件对象。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWUnregisterTimer(
HCHANNEL	hChannel,
HTIMER	hTimer
);	

参数

hChannel [in]
 已经被打开的视频通道句柄。

hTimerEvent[in]
 已经被注册的时间事件对象的句柄。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.38 MWScheduleTimermethod

该函数用于调度时间事件对象。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWScheduleTimer(
HCHANNEL	hChannel,
HTIMER	hTimer,
LONGLONG	llExpireTime
);	

参数

hChannel [in]
 已经被打开的视频通道句柄。

hTimerEvent[in]
 已经被注册的时间事件对象。

llExpireTime [in]
 设定调度时间事件的时间值，该时间值是采集设备时钟的绝对时间值。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

当调度时间到临时，时间事件对象关联的事件就会被设置为有信号状态。

3.2.39 MWRegisterNotifymethod

该函数用于注册一个通知事件对象。

函数原形

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

参数

- hChannel [in]
 已经被打开的视频通道句柄。
- hNotifyEvent[in]
 一个事件句柄，它需要使用 Win32 的函数 CreateEvent()进行创建。
- dwEnableBits[in]
 通知的类型掩码，参见备注的介绍。

返回值

调用成功，返回通知事件对象的句柄，否则返回 NULL。

备注

dwEnableBits 可以被设置为以下一个或多个通知类型。

MWCAP_NOTIFY_INPUT_SOURCE_CHANGE	当输入信号发生改变时。
MWCAP_NOTIFY_INPUT_FORMAT_CHANGE	当输入信号的格式发生改变时。
MWCAP_NOTIFY_VIDEO_FIELD_INPUT	当输入信号的状态发生改变时。
MWCAP_NOTIFY_VIDEO_FRAME_BUFFERED	当输入信号的采集缓冲区已填充完毕时。
MWCAP_NOTIFY_VIDEO_INPUT_RESET	当输入信号的采集操作被重置时。

当设置的通知类型发生时，通知事件对象关联的事件就会被设置为有信号状态。

3.2.40 MWUnregisterNotifymethod

该函数用于注销一个通知事件对象。

函数原形

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

参数

- hChannel [in]
 已经被打开的视频通道句柄。
- hNotify[in]
 已经被注册的通知事件对象的句柄。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.41 MWGetNotifyStatusmethod

该函数用于返回当前的通知类型。

函数原形

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

参数

- hChannel [in]
 已经被打开的视频通道句柄。
- hNotify[in]
 已经被注册的通知事件对象的句柄。
- pullStatus [out]

返回通知类型，该值参见 MWRegisterNotify 函数的备注。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.42 MWStartVideoCapturemethod

启动通道的视频采集。

函数原形

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

参数

hChannel [in]

已经被打开的视频通道句柄。

hEvent[in]

一个事件句柄，它需要使用 Win32 的函数 CreateEvent()进行创建。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.43 MWStopVideoCapturemethod

停止通道的视频采集。

函数原形

```
MW_RESULT
```

LIBMWCAPTURE_API	
MWStopVideoCapture(
HCHANNEL	hChannel
);	

参数

hChannel [in]
已经启动视频采集的通道句柄。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.44 MWPinVideoBuffermethod

锁定一段虚拟内存，可以降低采集视频数据到这段内存时 CPU 占用。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

pbFrame [in]
虚拟内存地址。

cbFrame [in]
虚拟内存大小

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无

3.2.45 MWUnpinVideoBuffermethod

解锁一段虚拟内存，与 MWPinVideoBuffer()对应使用。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

pbFrame [in]
虚拟内存地址。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无

3.2.46 MWCaptureVideoFrameToVirtualAddressmethod

采集视频帧数据到系统内存中。

函数原形

```
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
);	

参数

- hChannel [in]
已经开启视频采集的通道句柄。
- iFrame[in]
待采集的视频帧序号。
- pbFrame [out]
用于存放采集数据的虚拟内存指针。
- cbFrame[in]
存放采集数据的内存的字节长度。
- cbStride[in]
存放采集数据的内存的步长。
- dwFOURCC[in]
采集视频帧的色彩空间格式,参考 MWFOURCC.h。
- cx [in]
采集视频帧的宽度。
- cy [in]
采集视频帧的高度。
- bBottomUp[in]
采集视频帧是否按从下到上来存放。
- pvContext[in]
用户自定义的上下文指针。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.47 MWCaptureVideoFrameToPhysicalAddressmethod

采集视频帧数据到内存物理地址中。

函数原形

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
);	

参数

- hChannel [in]
已经被开启视频采集的视频通道句柄。
- iFrame[in]
待采集的视频帧序号。
- lFrameAddress[in]
存放视频帧数据的物理内存地址。
- cbFrame[in]
存放采集数据的内存的字节长度。
- cbStride[in]
存放采集数据的内存的步长。
- dwFOURCC[in]
采集视频帧的色彩空间格式,参考 MWFOURCC.h。
- cx [in]
采集视频帧的宽度。
- cy [in]
采集视频帧的高度。
- bBottomUp[in]
采集视频帧是否按从下到上来存放。
- pvContext[in]
用户自定义的上下文指针。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.48 MWCaptureVideoFrameWithOSDToVirtualAddressmethod

采集视频帧数据到虚拟内存中。

函数原形

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
);	

参数

- hChannel [in]
已经开启视频采集的通道句柄。
- iFrame[in]
待采集的视频帧序号。
- pbFrame [out]
用于存放采集数据的虚拟内存指针。
- cbFrame[in]
存放采集数据的内存的字节长度。
- cbStride[in]
存放采集数据的内存的步长。
- bBottomUp[in]
采集视频帧是否按从下到上来存放。
- pvContext[in]
用户自定义的上下文指针。
- dwFOURCC[in]
采集视频帧的色彩空间格式,参考 MWFOURCC.h。
- cx [in]
采集视频帧的宽度。
- cy [in]
采集视频帧的高度。

pOSDImage[in]
OSD 图像的句柄，使用 MWCreateImage 获得。

pOSDRects[in]
OSD 图像的目标区域。

cOSDRects[in]
OSD 图像的目标区域的个数。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.49 MWCaptureVideoFrameWithOSDToPhysicalAddressmethod

采集视频帧数据到物理内存中。

函数原形

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
);	

参数

hChannel [in]
已经开启视频采集的通道句柄。

iFrame[in]
待采集的视频帧序号。

llFrameAddress[in]
存放视频帧数据的物理内存地址。

cbFrame[in]
存放采集数据的内存的字节长度。

cbStride[in]
存放采集数据的内存的步长。

bBottomUp[in]
采集视频帧是否按从下到上来存放。

pvContext[in]
用户自定义的上下文指针。

dwFOURCC[in]
采集视频帧的色彩空间格式,参考 MWFOURCC.h。

cx [in]
采集视频帧的宽度。

cy [in]
采集视频帧的高度。

pOSDImage[in]
OSD 图像的句柄,使用 MWCreateImage 获得。

pOSDRects[in]
OSD 图像的目标区域。

cOSDRects[in]
OSD 图像的目标区域的个数。

返回值

返回 MW_RESULT 值,可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.50 MWCaptureVideoFrameToVirtualAddressExmethod

采集视频帧数据到系统内存中。

函数原形

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
);	

参数

hChannel [in]

已经开启视频采集的通道句柄。

iFrame[in]

待采集的视频帧序号。

pbFrame [out]

用于存放采集数据的虚拟内存指针。

cbFrame[in]

存放采集数据的内存的字节长度。

cbStride[in]

存放采集数据的内存的步长。

bBottomUp[in]

采集视频帧是否按从下到上来存放。

pvContext[in]

用户自定义的上下文指针。

dwFOURCC[in]

采集视频帧的色彩空间格式,参考 MWFOURCC.h。

cx [in]

采集视频帧的宽度。

cy [in]
采集视频帧的高度。

dwProcessSwitchs[in]
视频处理的掩码值。参考 MWCAP_VIDEO_PROCESS_xx。

cyPartialNotify[in]
使用分行采集时的行数。

pOSDImage[in]
OSD 图像的句柄，使用 MWCreateImage 获得。

pOSDRects[in]
OSD 图像的目标区域。

cOSDRects[in]
OSD 图像的目标区域的个数。

sContrast[in]
对比度。

sBrightness[in]
亮度。

sSaturation[in]
饱和度。

sHue[in]
色度。

deinterlaceMode[in]
去隔行模式。

aspectRatioConvertMode[in]
宽高比转换模式。

pRectSrc [in]
采集画面的源区域。

pRectDest[in]
采集画面的目标区域。

nAspectX[in]
宽高比中的宽度。

nAspectY[in]
宽高比中的高度。

colorFormat[in]
色彩格式标准。

quantRange[in]
量化范围。

satRange[in]
饱和度范围。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW _SUCCEEDED	调用成功。
MW _FAILED	调用失败。

MW_INVALID_PARAMS | 输入的参数有错误。

备注

无。

3.2.51 MWCaptureVideoFrameToPhysicalAddressExmethod

采集视频帧数据到物理内存中。

函数原形

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
);	

参数

hChannel [in]
已经开启视频采集的通道句柄。

iFrame[in]
待采集的视频帧序号。

llFrameAddress[in]
存放视频帧数据的物理内存地址。

cbFrame[in]
存放采集数据的内存的字节长度。

cbStride[in]
存放采集数据的内存的步长。

bBottomUp[in]
采集视频帧是否按从下到上来存放。

pvContext[in]
用户自定义的上下文指针。

dwFOURCC[in]
采集视频帧的色彩空间格式,参考 MWFOURCC.h。

cx [in]
采集视频帧的宽度。

cy [in]
采集视频帧的高度。

dwProcessSwitchs[in]
视频处理的掩码值。参考 MWCAP_VIDEO_PROCESS_xx。

cyPartialNotify[in]
部分通知的个数。

pOSDImage[in]
OSD 图像的句柄,使用 MWCreateImage 获得。

pOSDRects[in]
OSD 图像的目标区域。

cOSDRects[in]
OSD 图像的目标区域的个数。

sContrast[in]
对比度。

sBrightness[in]
亮度。

sSaturation[in]
饱和度。

sHue[in]
色度。

deinterlaceMode[in]
去隔行模式。

aspectRatioConvertMode[in]
宽高比转换模式。

pRectSrc [in]
采集画面的源区域。

pRectDest[in]
采集画面的目标区域。

nAspectX[in]
宽高比中的宽度。

nAspectY[in]
宽高比中的高度。

colorFormat[in]
色彩格式标准。

quantRange[in]
量化范围。

satRange[in]
饱和度范围。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.52 MWGetVideoBufferInfomethod

获取视频缓冲信息。

函数原形

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

参数

hChannel [in]
已经开启视频采集的通道句柄。

pVideoBufferInfo [out]
视频缓冲信息。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。

MW_INVALID_PARAMS | 输入的参数有错误。

备注

无。

3.2.53 MWGetVideoFrameInfomethod

获取视频帧信息。

函数原形

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

参数

- hChannel [in]
已经开启视频采集的通道句柄。
- i [in]
视频帧的序号。
- pVideoFrameInfo [out]
返回视频帧的信息。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.54 MWGetVideoCaptureStatusmethod

获取视频采集状态。

函数原形

```
MW_RESULT
LIBMWCAPTURE_API
MWGetVideoCaptureStatus(
    HCHANNEL                hChannel,
```

```
MWCAP_VIDEO_CAPTURE_STATUS * pStatus
);
```

参数

hChannel [in]
已经开启视频采集的通道句柄。

pStatus [out]
返回视频采集状态。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.55 MWStartAudioCapturemethod

开始该通道的音频采集。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.56 MWStopAudioCapturemethod

停止该通道的音频采集。

函数原形

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

参数

hChannel [in]
已经开启音频采集的通道句柄。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.57 MWCaptureAudioFrameMethod

采集一帧音频数据。

函数原形

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

参数

hChannel [in]
已经开启音频采集的通道句柄。
pAudioCaptureFrame [out]
返回一帧音频。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。

MW_INVALID_PARAMS | 输入的参数有错误。

备注

无。

3.2.58 MWSetVideoInputAspectRatiomethod

设置视频输入信号的宽高比。

函数原形

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

参数

- hChannel [in]
 已经打开的通道句柄。
- nAspectX [in]
 指定宽高比中的宽度。
- nAspectY [in]
 指定宽高比中的高度。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.59 MWGetVideoInputAspectRatiomethod

获取视频输入信号的宽高比。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWGetVideoInputAspectRatio(
HCHANNEL	hChannel,

int*	pnAspectX,
int*	pnAspectY
);	

参数

hChannel [in]
已经打开的通道句柄。

nAspectX [out]
返回宽高比中的宽度。

nAspectY [out]
返回宽高比中的高度。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.60 MWSetVideoInputColorFormatmethod

设置视频输入的颜色格式标准。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWSetVideoInputColorFormat(
HCHANNEL	hChannel,
MWCAP_VIDEO_COLOR_FORMAT	colorFormat
);	

参数

hChannel [in]
已经打开的通道句柄。

colorFormat [in]
颜色格式标准。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.61 MWGetVideoInputColorFormatmethod

获取视频输入的颜色格式标准。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

pColorFormat [out]
返回颜色格式标准。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.62 MWSetVideoInputQuantizationRangemethod

设置视频输入的量化范围。

函数原形

```
MW_RESULT
LIBMWCAPTURE_API
MWSetVideoInputQuantizationRange(
    HCHANNEL                hChannel,
    MWCAP_VIDEO_QUANTIZATION_RANGE quantRange
);
```

参数

hChannel [in]
已经打开的通道句柄。

quantRange [in]
量化范围。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.63 MWGetVideoInputQuantizationRangemethod

获取视频输入的量化范围。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

pQuantRange [in]
返回量化范围。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.64 MWSetLEDModemethod

设置 LED 模式。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

dwMode [in]
LED 模式。参考 MWCAP_LED_MODE。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.65 MWGetFirmwareStorageInfomethod

获取固件存储信息。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

pFirmwareStorageInfo [out]
返回固件存储信息。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.66 MWEraseFirmwareDatamethod

擦除固件数据。

函数原形

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

参数

hChannel [in]

已经打开的通道句柄。

cbOffset [in]

要擦除的数据的起始位置。

cbErase [in]

要擦除的数据的长度。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.67 MWReadFirmwareDatamethod

读取固件数据。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	

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

参数

hChannel [in]

已经打开的通道句柄。

cbOffset [in]

要读取的数据的起始位置。

pbyData [out]

返回读取的数据。

cbToRead [in]

要读取的数据的长度。

pcbRead [out]

返回读取的实际长度。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.68 MWWriteFirmwareData方法

写固件数据。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWWriteFirmwareData(
HCHANNEL	hChannel,
DWORD	cbOffset,
BYTE *	pbyData,
DWORD	cbData
);	

参数

hChannel [in]
已经打开的通道句柄。

cbOffset [in]
要写入的数据的起始位置。

pbyData [in]
要写入的数据。

cbData [in]
要写入的数据长度。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.69 MWSetPostReconfigmethod

发送重读配置的指令，延迟执行。

函数原形

MW_RESULT	
LIBMWCAPTURE_API	
MWSetPostReconfig(
HCHANNEL	hChannel,
DWORD	dwDelayMS
);	

参数

hChannel [in]
已经打开的通道句柄。

dwDelayMS [in]
延迟执行的时间，单位毫秒。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.70 MWCreateImagemethod

创建 OSD 图像。

函数原形

HOSD
LIBMWCAPTURE_API
MWCreateImage(
 HCHANNEL
 int
 int
);

hChannel,
cx,
cy

参数

hChannel [in]
 已经打开的通道句柄。

cx [in]
 图像的宽度。

cy [in]
 图像的高度。

返回值

返回 图像句柄。如果失败返回 NULL。

备注

无。

3.2.71 MWOpenImagemethod

打开图像。
Open image.

函数原形

MW_RESULT
LIBMWCAPTURE_API
MWOpenImage(
 HCHANNEL
 HOSD
 LONG*
);

hChannel,
hImage,
plRet

参数

hChannel [in]

已经打开的通道句柄。

hImage [in]
图像句柄。

plRet [out]
返回该图像的引用计数。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.72 MWCloseImagemethod

关闭图像。

函数原形

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

参数

hChannel [in]
已经打开的通道句柄。

hImage [in]
图像句柄。

plRet [out]
返回该图像的引用计数。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.73 MWUploadImageFromVirtualAddressmethod

从系统内存上载图像到采集设备。

函数原形

```
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
);
```

参数

- hChannel [in]
已经打开的通道句柄。
- hImage [in]
图像句柄。
- cfDest [in]
目标图像的色彩格式标准。
- quantRangeDest [in]
目标图像的量化范围。
- satRangeDest [in]
目标图像的饱和度范围。
- xDest [in]
水平方向的目标位置。
- yDest [in]
垂直方向的目标位置。
- cxDest [in]

目标图像的宽度。

cyDest [in]
目标图像的高度。

pvSrcFrame [in]
源图像数据。

cbSrcFrame [in]
源图像数据长度。

cbSrcStride [in]
源图像数据步长。

cxSrc [in]
源图像的宽度。

cySrc [in]
源图像的高度。

bSrcBottomUp [in]
源图像是否上下颠倒的。

bSrcPixelAlpha [in]
源图像中的像素是否具有 Alpha 分量。

bSrcPixelXBGR [in]
源图像中的像素的颜色格式是否是 XBGR。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

3.2.74 MWUploadImageFromPhysicalAddressmethod

从物理内存上载图像到采集设备。

函数原形

```
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
);	

参数

hChannel [in]
已经打开的通道句柄。

hImage [in]
图像句柄。

cfDest [in]
目标图像的色彩格式标准。

quantRangeDest [in]
目标图像的量化范围。

satRangeDest [in]
目标图像的饱和度范围。

xDest [in]
水平方向的目标位置。

yDest [in]
垂直方向的目标位置。

cxDest [in]
目标图像的宽度。

cyDest [in]
目标图像的高度。

lSrcFrameAddress [in]
源图像数据所存储的物理地址。

cbSrcFrame [in]
源图像数据长度。

cbSrcStride [in]
源图像数据步长。

cxSrc [in]
源图像的宽度。

cySrc [in]
源图像的高度。

bSrcBottomUp [in]
源图像是否上下颠倒的。

bSrcPixelAlpha [in]

源图像中的像素是否具有 Alpha 分量。

bSrcPixelXBGR [in]

源图像中的像素的颜色格式是否是 XBGR。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

.3.2.75 MWCreateExtendObjectmethod

获取通道的扩展信息。

函数原形

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

参数

hChannel [in]

已经打开的通道句柄。

guid [in]

扩展信息标志位。

ppObject [out]

扩展信息结构体的大小。

返回值

返回 MW_RESULT 值，可能的返回值如下表所示

MW_SUCCEEDED	调用成功。
MW_FAILED	调用失败。
MW_INVALID_PARAMS	输入的参数有错误。

备注

无。

4 联系我们

请访问以下网站以获取最新版本：

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

如有任何问题，请邮件联系 support@magewell.net