

USBOSDM2 SDK User Manual

Writing Software to Control Multi-Channel Video Input, OSD & MPEG Compression on USB Encoder

Version 1.0.0

Copyright © 2010 [Inventa Australia Pty Ltd](#)

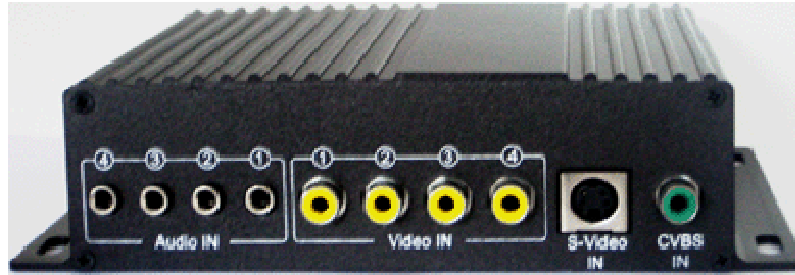


Table of Contents

1. Introduction	-----	2
2. Device Architecture	-----	2
3. USBOSDM2.DLL and USBOSDM2 SDK	-----	3
4. SDK Function Details	-----	3
■ SDK Start & Stop Functions	-----	3
■ Generic USBOSDM2 Functions	-----	5
■ OSD Device Functions	-----	6
- Video Input Channel Functions	-----	6
---- Channel TV Signal Functions	-----	6
---- Channel Size & Location Functions	-----	8
---- Channel Cropping Functions	-----	8
---- Channel Boundary Functions	-----	9
---- Channel Mirroring Functions	-----	10
---- Zooming Functions	-----	10
---- Channel Video Colour Functions	-----	11
---- Channel Background Colour Functions	-----	13
- Audio Input Channel Functions	-----	14
- Overlay Functions	-----	15
---- Bitmap Graphics Overlay Functions	-----	15
---- Single Box Overlay Functions	-----	21
- Generic OSD Device Functions	-----	24
■ M2B Device Functions	-----	25
- M2B Video Input Functions	-----	25
- M2B Video Preview Functions	-----	28
- M2B Audio Preview Functions	-----	30
- MPEG Encoding Functions	-----	30
- Video File Recording Functions	-----	35
- Video Streaming Functions	-----	37
- Still Image Grabbing Functions	-----	38
- Show Status Functions	-----	38
- Generic M2B Device Functions	-----	40
5. SDK Functions Calling Sequence	-----	40
6. Using DirectShow Filters with the SDK	-----	41
7. SDK Installation & Running Environment	-----	41
■ Install the SDK	-----	41
■ Create Application with the SDK	-----	41
8. Sample Source Codes	-----	42
9. Hardware Specification	-----	42

1. Introduction

USBOSDM2 SDK (Software Development Kit) is for writing customized application software to control **USBOSDM2** --- a USB-powered multi-channel A/V Input and mixing, colour OSD and hardware MPEG encoding device on PC. **USBOSDM2 SDK** shields the programmer from interfacing the complicated hardware registers on board the PCB, or programming the DirectShow filters directly, by using simple function calls to accomplish complicated tasks, such as showing mouse-resizable floating video preview window on PC screen, mixing multiple colour overlay items on MPEG video, etc.

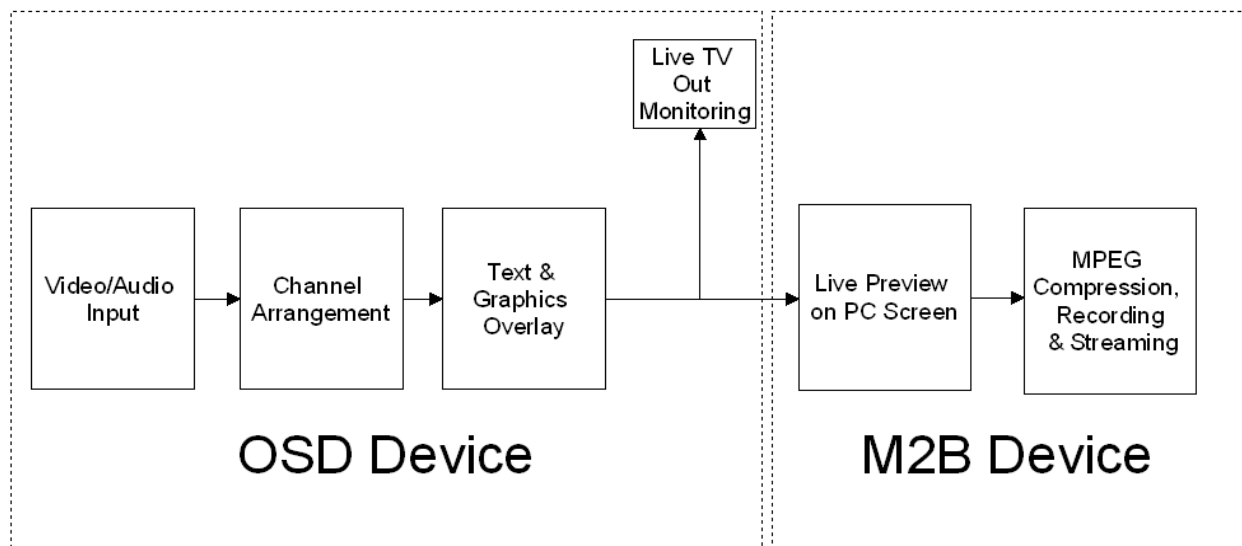
USBOSDM2 has some major functions and capabilities as listed below:

- Real-time encode MPEG1, MPEG2 video up-to 25 Mbps using hardware compression chipset
- Encode 4 video input into one MPEG stream as picture-in-picture in arbitrary position and size
- Encode unlimited colour graphics and overlay (OSD) box overlay as integral part of the MPEG video
- Fully USB-Powered, no AC adaptor is needed
- Real-time live preview video on PC screen in arbitrarily re-sizable window or full screen
- 4 Video Input Channels software selectable from 5 Composite and 1 SVideo Sockets
- 4 Audio Input Channels software selectable from 4 Stereo Line-in and 2 Stereo Mic. input sockets
- Perfect Audio/Video Sync is always maintained in the recorded and streamed MPEG video
- Record video as MPEG files: files can be manually split or automatically split at fixed time or length
- Record using timer or calendar scheduler with daily or weekly repetition option
- Real-time stream video over IP network multi-cast or uni-cast independent of file recording status
- Real-time flip/mirror any input video horizontally or vertically
- Real-time enlarge input video at any point inside video frame (2-times zoom-in)
- PAL and NTSC encoding at 720X576, 720X480, 480X576, 480X480, 352X288, 352X240-Pixels
- DVD, SVCD, VCD, MPEG2, MPEG1 encoding format fully user-selectable
- Up to 8 USBOSDM2 devices can run simultaneously on one PC under the same software
- Bitmap graphics colour overlay on video with alpha and blink options
- Box overlay on video with colour, alpha and boundaries
- Each input video channel can have boundary on or off
- Each input video channel has colour brightness, contrast, hue, saturation and sharpness control
- Each input audio channel has hardware gain control and left/right mute
- Recording file name can have user-defined fields inc. recording quality, date/time, serial number etc
- Still images can be grabbed in bmp, jpg, gif, tiff, and png format
- Live recording status can be displayed inside video frame with user-definable colour and font
- Colour Bar generation to test video output capability
- Can record/stream MPEG Video with Text/Graphics Overlay when no input signal is available
- Device Driver available for MS Windows 7, Vista and XP

Using **USBOSDM2 SDK**, complicated application software can be written quickly in any high-level programming languages such as C, C++, VisualBasic, C#, Delphi, etc., to fully control the **USBOSDM2** hardware realizing all the functions as implemented by the **Application Software** bundled with **USBOSDM2** device, plus more possible functions supported by the hardware but not implemented by the bundled **Application Software**.

2. Device Architecture

USBOSDM2 hardware consists of an **OSD** device and an **M2B** device: the **OSD** device is responsible for mixing input channel signals, arranging channel video windows' location and size, overlaying text/graphics on the video surface, and outputting such mixed and overlaid video to the TV Out sockets and the **M2B** device. The **M2B** device receives the video signal from **OSD** device's output, displays the un-compressed Video on PC screen as live preview, compresses it into MPEG data stream, records the MPEG data onto disk files, and streams the MPEG data over network. The SDK follows this two-device architecture and its main function is to send controlling commands to the **OSD** and **M2B** devices.



USBOSDM2 Device Architecture

3. USBOSDM2.dll and USBOSDM2 SDK

USBOSDM2 SDK is based on a dynamic linking library **USBOSDM2.dll** which supports all the function calls application software might use, and several supporting libraries. **USBOSDM2.dll** and these libraries need to be installed on the PC before the application software can use the SDK.

4. SDK Function Details

Note the “Output Parameters” in any following functions will only be set properly when those function calls succeed. If a function call failed, the value of any Output Parameter will be uncertain.

4.1 SDK Start and Stop Functions

```

USBOSDM2_API int USBOSDM2_initDevs( HWND &SDKCtrlWnd,
                                     HWND &SDKVideoWnd,
                                     HWND parentWnd = NULL,
                                     char *m2bList = NULL,
                                     HWND *ownerWnd = NULL,
                                     RECT *posRect = NULL,
                                     bool useDefaultSettings = false,
                                     bool loadFirmware = true,
                                     bool noOSDDevice = false);
  
```

Function: Usually the first function to call when using the SDK: it initializes all USBOSDM2 devices and returns totalDevs: the total number of USBOSDM2 devices connected to the PC. On failure this function returns 0. Most **USBOSDM2 SDK** functions can only work after **USBOSDM2_initDevs()** returns > 0.

Output Parameter SDKCtrlWnd: Handle of the Control Window of USBOSDM2 SDK: this will always be a valid window handle on a successful call to **USBOSDM2_initDevs**, even when it was made invisible through function **USBOSDM2_controlWinOn**.

Output Parameter SDKVideoWnd: Handle of the Video Window of USBOSDM2 SDK: this will always be a valid window handle on a successful call to **USBOSDM2_initDevs**. This window will serve as the SDK’s default Video Window within which all **M2B** devices will display their live video preview. If the user application software wishes to use its own video window to display live video preview this default video window usually need to be made invisible by calling function **USBOSDM2_videoWinOn**.

Please Note: Application Software can send normal Windows messages such as WM_MOVE to SDKCtrlWnd and SDKVideoWnd, or call Windows SDK functions such as ShowWindow() on these two Window handles, but Application Software Should NOT

call Windows' functions or send Windows' messages to close or destroy these two windows: Application Software should always call USBOSDM2_endDevs() function to terminate USBOSDM2 SDK which will properly shutdown these two windows after releasing all resources.

Input Parameter parentWnd: Handle of a window created by the application software: this window will receive a WM_CLOSE message from USBOSDM2 SDK if user chooses to exit the application from within the USBOSDM2 SDK (e.g. user selected "Exit Program" from the Drop-down Menu): If this parameter is NULL then there will be no message passing back from USBOSDM2 SDK to user's application software that calls functions of the SDK.

As defined in the header file USBOSDM2.h, the SDK could also send these user-defined **USBOSDM2_MSG_XXXX** messages to this window:

```
#define USBOSDM2_MSG_ABOUT_DIALOG    WM_USER + 50 // The SDK's "About..."  
Menu Item was selected by end-user  
#define USBOSDM2_MSG_SDK_EXIT        WM_USER + 51 // The SDK has exited
```

Where MS Windows' SDK usually defines WM_USER as 0x0400.

When receiving **USBOSDM2_MSG_SDK_EXIT**, the application software can assume the **USBOSDM2** SDK has ended through end-user selecting "Exit" menu etc. mechanism from the SDK's default Video Window or Control Window.

Output Parameter m2bList: Array of maximum 8 members: each member represents an M2B device whose enumeration number (in the order the device is found) is this member's index in m2bList array. On return, each member of m2bList indicates if the video for this USBOSDM2 device has been started: non-zero means video window for that device is started, zero means not started.

If m2bList is NULL, no returned info will be available on if each M2B device has been started.

Input Parameter ownerWnd: An array of length not smaller than the total found USBOSDM2 devices in the PC. When ownerWnd is non-NULL, its i'th member indicates the owner window of the i'th M2B device, i = 0 ~ (totalDevs - 1). An M2B device's owner window is where the video display will appear for that M2B device. If ownerWnd is non-NULL but a member of ownerWnd is NULL then the corresponding M2B device's video display will use Windows' Desktop Window as its owner window. If ownerWnd itself is NULL (the default situation) then all M2B devices will use SDKVideoWnd as their owner window: all M2B devices' video display will be equally spread across SDKVideoWnd Window horizontally, all video's height will be the same as SDKVideoWnd Window's height.

Input Parameter posRect: This is only meaningful when ownerWnd is not NULL: each member indicates the upper left corner (x, y) and width/height of the video appearing inside its owner window. If a member of posRect is NULL then that M2B device's video will occupy the entire client area of its owner window. If posRect itself is NULL then every M2B device's video will occupy its owner window's entire client area.

Input Parameter useDefaultSettings: If set to TRUE the SDK will use default values -- as described in the "**USBOSDM2 Application Software User Manual**" -- for all its user-definable variables, ignoring the USBOSDM2.ini file's contents saved when last time the SDK exit. This parameter default to FALSE, meaning when starting up the SDK will use the saved USBOSDM2.ini file contents under the folder where USBOSDM2.dll resides to set values to all variables.

The USBOSDM2.ini file does not exist on a PC when the USBOSDM2 SDK is first time used, but will be created each time the USBOSDM2.DLL exit.

Input Parameter loadFirmware: TRUE (default) means firmware file "USBOSD.bix" will be loaded onto all OSD devices.

Input Parameter noOSDDevice: Should always set to false. True means debugging purpose to disable OSD device on start.

Return: Total number of **USBOSDM2** Devices installed in the PC. 0 means no device.

Note 1: Each connected **USBOSDM2** device must have its two drivers installed properly to be counted by this function.
Note 2: Depending on the number of installed devices (maximum 8) this function can take some time to return.
Note 3: Every **USBOSDM2_initDevs()** call must have a matching **USBOSDM2_endDevs** (see below) call.

USBOSDM2_API void USBOSDM2_endDevs(void);

Function: End **USBOSDM2** SDK. If the application has called **USBOSDM2_initDevs()**, it must call **USBOSDM2_endDevs()** before exiting, regardless the return value from **USBOSDM2_initDevs()**.

Note: A second call to **USBOSDM2_initDevs()** without calling **USBOSDM2_endDevs** to end the first **USBOSDM2_initDevs()** call should **never** be attempted – this can cause memory leaking and process hanging!

4.2 Generic USBOSDM2 Functions

USBOSDM2_API int USBOSDM2_totalDevs(void);

Function: Return the total number of **USBOSDM2** devices obtained from calling **USBOSDM2_initDevs()** without re-initializing the hardware. If **USBOSDM2_initDevs()** has never been called this function will return 0.

USBOSDM2_API void USBOSDM2_getSDKVer(char * ver);

Function: Copy current **USBOSDM2** SDK's version number into returned variable "ver" as NULL-terminated string.

Output Parameter ver: Must point to a memory area long enough to hold 10 or more bytes. First version is "1.0.0".

USBOSDM2_API void USBOSDM2_setSoftwareName(char *name);

Function: Assign user-defined name to the SDK so that all message prompt/dialog display use this name to refer to the application software. If not set, the default program name is "USBOSDM2".

Input Parameter name: The string used as the software name. If this parameter is NULL then this function has no effect.

USBOSDM2_API unsigned char USBOSDM2_getHardwareVer(int devNum);

Function: Return the **USBOSDM2** device's Hardware Version number.

Input Parameter devNum: must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Return Value: 0xFF for failure, other values are Hardware Version Number, 1 is the first version.

USBOSDM2_API void USBOSDM2_setConfirmOnExit(bool confirm);

Function: Decide if to display a confirmation window when **USBOSDM2** DLL is to exit, if the SDK's Control Window or Video Window is shown (i.e., **USBOSDM2_controlWinOn(true)** or **USBOSDM2_videoWinOn(true)** was previously called).

Input Parameter confirm: True (default) will display the confirmation dialog, false will not display therefore exit will happen without user intervention.

Note confirming dialog can only appear if the SDK's Control or Video Window is shown.

USBOSDM2_API bool USBOSDM2_getConfirmOnExit(void);

Function: Return if confirming dialog will be displayed when the SDK is to exit: TRUE means display a dialog. Note the confirming dialog can only appear if the SDK's Control Window or Video Window is shown.

USBOSDM2_API bool USBOSDM2_setVideoWinIsChild(bool isChild);

Function: Indicate to the SDK that its default Video Window is a Child Window or not.

Input Parameter isChild: True means the Video Window is child window, false (default) means it's a top level window.

Return: The previous status of the Video Window: true means it was child window, false means it was top level window.

Note: Call this function with isChild=True to hide the menu items Maximize, Exit and About from the **Drop Down Menu**: this is necessary when the 3rd parameter (parentWnd) passed to **USBOSDM2_initDevs** is a valid window handle the application software intends to use as the parent window for the SDK's default Video Window.

USBOSDM2_API void USBOSDM2_setExitOnTimerEnd(bool exit);

Function: Decide if to exit the **USBOSDM2** SDK when a recording timer expires.

Input Parameter exit: True means the SDK will exit when a recording ends due to the recording timer expires.

Note 1: This option is false by default.

Note 2: When this option is true and a recording timer expires, SDK exit will not happen if some other USBOSDM2 device is still recording, or some dialog boxes such as the “Device Setup” etc. opened from inside the SDK are not closed.

USBOSDM2_API bool USBOSDM2_getExitOnTimerEnd(void);

Function: Return if the USBOSDM2 SDK will exit when a recording timer expires.

USBOSDM2_API void USBOSDM2_setShutdownPCOnTimerEnd (bool shutdown);

Function: Decide if to shutdown the PC when a recording timer expires.

Input Parameter shutdown: True means the SDK will shutdown PC when a recording ends due to the recording timer expires.

Note 1: This option is false by default.

Note 2: When this option is true and a recording timer expires, shutdown PC will not happen if any USBOSDM2 device is still recording, or some dialog boxes such as the “Device Setup” etc. opened from inside the SDK are not closed.

USBOSDM2_API bool USBOSDM2_getShutdownPCOnTimerEnd (void);

Function: Return if the USBOSDM2 SDK will shutdown PC when a recording timer expires.

USBOSDM2_API void USBOSDM2_firmwareFile(char *fileName);

Function: Set default firmware file name to fileName for all OSD Devices to use.

Input Parameter filename: Name of a firmware file for OSD device.

Note: SDK provided default firmware file is USBOSD.bix.

4.3 OSD Device Functions

4.3.1 Video Input Channel Functions

---- Channel TV Signal Functions

USBOSDM2_API bool USBOSDM2_setChanAutoDetectTVType(int devNum, int chanNum, bool enable = true);

Function: Enable USBOSDM2 Device's auto detect on a channel's incoming signal type (PAL/NTSC/SECAM).

Input Parameter devNum: OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter enable: True (default) for enabling auto detect, false for disabling auto detect.

Return: True for success.

Note: Although by default the OSD Device is in Auto Detect mode, the application software can always call this function on all channels to make sure they are auto-detecting incoming signal's TV type.

USBOSDM2_API bool USBOSDM2_setChanVideoSrc(int devNum, unsigned short chanNum, int src);

Function: Set input channel's video source socket.

Input Parameter devNum: OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter src: 1 for RCA socket of input 1, 2 for RCA socket of input 2,
3 for RCA socket of input 3, 4 for RCA socket of input 4,
5 for RCA socket of input 5, 6 for SVideo socket of input 5.

Return: True for success.

USBOSDM2_API int USBOSDM2_getChanVideoSrc(int devNum, unsigned short chanNum);

Function: Get input channel's video source socket.

Input Parameter devNum: OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Return: 1 for RCA socket of input 1, 2 for RCA socket of input 2,
3 for RCA socket of input 3, 4 for RCA socket of input 4,
5 for RCA socket of input 5, 6 for SVideo socket of input 5,
0 for failure.

USBOSDM2_API int USBOSDM2_detectAndSetTVType(int devNum, int & chanNum);

Function: If any channel of the OSD Device has signal, return the TV Format (0 for NTSC, 1 for PAL) of the 1st socket having signal, else return 2 (no channel has signal).
This function set OSD Device's Video Output TV Format and M2B's Video Input TV Format according to:
(1) The TV format of the 1st input socket that has TV signal
(2) If no socket has signal, set the OSD Device Video Output TV Format according to OSD Device's defaultTVFormat value (either 0 for NTSC, or 1 for PAL)

Input Parameter devNum: OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter chanNum: If any channel has signal, the channel number of the first channel having signal (0~3).

Return: 0 for NTSC at the first channel having signal
1 for PAL at the first channel having signal
2 for no channel has signal
-1 for failure.

USBOSDM2_API int USBOSDM2_getSignalType(int devNum, int socket);

Function: Get USBOSDM2 device's input socket signal type

Input Parameter devNum: **OSD Device** number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter socket: **OSD Device input socket** number, must be between [0, 4] inclusive.

Return: the channel signal type.

When socket is between [0, 3], return signal type:

0 = NTSC(M)
1 = PAL (B,D,G,H,I)
2 = SECAM
3 = NTSC4.43
4 = PAL (M)
5 = PAL (CN)
6 = PAL 60

When socket is 4, return signal type:

0 NTSC M
1 PAL (B, G, H, I, N)
2 PAL M
3 Combination N
4 NTSC 4.43

Returns -1 for failure.

USBOSDM2_API int USBOSDM2_getSignalAvailable(int devNum, int socket);

Function: Get OSD Device channel's TV signal availability.

Input Parameter devNum: **OSD Device** number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter socket: **OSD Device Input socket** number, must be between [0, 4] inclusive.

Return: TV_PAL(1) for 50Hz signal type, TV_NTSC(0) for 60Hz signal type,
TV_NONE(2) for no signal at this channel, -1 for failure.

USBOSDM2_API void USBOSDM2_setDefaultTVType(int devNum, int tvtype);

Function: Set the default TV Signal Type of USBOSDM2 to either PAL (1) or NTSC (0).

Input Parameter devNum: **OSD Device** number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter tvtype: must be either 1 for PAL(50Hz) or 0 for NTSC(60Hz).

Note 1: Default TV Signal Type is used to set video output and MPEG encoding parameters when all 4 input channels of the OSD device have no video signals available.

Note 2: When some input channels have video signal available, the default TV signal type will be automatically set to the type of the video signal of the first input channel that has signal.

---- Channel Size & Location Functions

```
USBOSDM2_API bool USBOSDM2_setChanPos(int devNum,  
                                         int chanNum,  
                                         unsigned short left,  
                                         unsigned short top,  
                                         unsigned short width,  
                                         unsigned short height);
```

Function: Set OSD Device channel video window's position and size inside output video frame.

Input Parameter devNum: **OSD Device** number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter left: the channel video window's left position inside output video frame, in pixel unit.

Input Parameter top: the channel video window's top position inside output video frame, in pixel unit.

Input Parameter width: the channel video window's width inside output video frame, in pixel unit.

Input Parameter height: the channel video window's height inside output video frame, in pixel unit.

Return: TRUE for success.

Note: For PAL video, channel video window position/size limit for [left, top, width, height] is [0, 0, 720, 576],

For NTSC video, channel video window position/size limit for [left, top, width, height] is [0, 0, 720, 480].

```
USBOSDM2_API bool USBOSDM2_getChanPos(int devNum,  
                                         int chanNum,  
                                         unsigned short &left,  
                                         unsigned short &top,  
                                         unsigned short &width,  
                                         unsigned short &height);
```

Function: Get OSD Device channel video window's position and size inside output video frame.

Input Parameter devNum: **OSD Device** number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter left: the channel video window's left position inside output video frame, in pixel unit.

Output Parameter top: the channel video window's top position inside output video frame, in pixel unit.

Output Parameter width: the channel video window's width inside output video frame, in pixel unit.

Output Parameter height: the channel video window's height inside output video frame, in pixel unit.

Return: TRUE for success.

---- Channel Cropping Functions

```
USBOSDM2_API bool USBOSDM2_setChanCrop  
    (int devNum, int chanNum, WORD left, WORD &top, WORD width, WORD &height);
```

Function: Set the cropping area for an OSD Device input video channel.

Input Parameter devNum: **OSD Device** number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter left: the channel left cropping value, in pixel unit, valid values are 1~143 for PAL, 1~137 for NTSC.

Input/Output Parameter top: the channel top cropping value, in pixel unit, is always 10 for PAL, 12 for NTSC.

Input Parameter width: the channel cropping width value, in pixel unit, maximum 720.

Input/Output Parameter height: the channel cropping height value, in pixel unit, maximum 576 for PAL, 480 for NTSC.

Return: TRUE for success.

Note 1: A video input channel's Cropping Area is the actual video pixel area fetched out of the raw video decoding frame after an **OSD Device** digitized the input analogue video signal: the raw video decoding frame is larger than standard 720X576-Pixel PAL and 720X576-Pixel NSTC video frame. The Cropping Area is limited to maximum 720X576-Pixel for PAL or 720X480-Pixel for NSTC video input, and is used for the actual video overlay, channel mixing and

MPEG encoding processing. The pixels on the raw video decoding frame outside the Cropping Area are abandoned for any processing.

Note 2: On return this function can change the “top” and “height” values of the cropping area to fit into the current TV signal type required cropping area.

Note 3: Default Cropping Area: for PAL: Left=15, Top=10, Width=720, Height=576
for NTSC: Left=15, Top=12, Width=720, Height=480

USBOSDM2_API bool USBOSDM2_getChanCrop

(int devNum, int chanNum, WORD &left, WORD &top, WORD &width, WORD &height);

Function: Get the cropping area for an OSD Device input video channel.

Input Parameter devNum: **OSD Device** number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter left: the channel left cropping value, in pixel unit.

Output Parameter top: the channel top cropping value, in pixel unit.

Output Parameter width: the channel cropping width value, in pixel unit.

Output Parameter height: the channel cropping height value, in pixel unit.

Return: TRUE for success.

---- Channel Boundary Functions

USBOSDM2_API bool USBOSDM2_enableChanBoundary(int devNum, int chanNum, bool enable);

Function: Enable/Disable Channel Boundary for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter enable: TRUE for enabling channel boundary, false for disabling boundary (default).

Return: True for success.

USBOSDM2_API bool USBOSDM2_getEnableChanBoundary(int devNum, int chanNum, int &enable);

Function: Get Channel Boundary enabling status for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter enable: non-zero for enabling channel boundary, zero for disabling.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setChanBoundaryColour(int devNum, unsigned char bc);

Function: Set Channel Boundary Colour for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter bc: 0=Black boundary; 1=25% Gray boundary; 2=72% Gray boundary; 3= 100% White boundary.

Return: True for success.

Note: Once set, all channels on an **OSD Device** will have the same boundary colour, default boundary colour is bc==3.

USBOSDM2_API bool USBOSDM2_getChanBoundaryColour(int devNum, unsigned char &bc);

Function: Get Channel Boundary Colour for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter bc: 0=Black boundary; 1=25% Gray boundary; 2=72% Gray boundary; 3= 100% White boundary.

Return: True for success.

---- Channel Mirroring Functions

USBOSDM2_API bool USBOSDM2_setChanMirror(int devNum, int chanNum, bool HoriMirror, bool VertMirror);

Function: Set One Channel's Video Mirroring for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter HoriMirror: True for enabling Horizontal Mirroring on the channel's video.

Input Parameter VertMirror: True for enabling Vertical Mirroring on the channel's video.

Return: True for success.

Note: Default is no mirroring on both directions.

USBOSDM2_API bool USBOSDM2_getChanMirror

(int devNum, int chanNum, bool &HoriMirror, bool &VertMirror);

Function: Get One Channel's Video Mirroring Status of an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter HoriMirror: True indicates Horizontal Mirroring is enabled for this channel.

Output Parameter VertMirror: True indicates Vertical Mirroring is enabled for this channel.

Return: True for success.

---- Zooming Functions

USBOSDM2_API bool USBOSDM2_setZoomEnable(int devNum, bool enable);

Function: Enable/Disable Video Zooming on an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter enable: True to enable the 2-times zooming (enlarge video by 2 times), false to disable zooming (default).

Return: True for success.

USBOSDM2_API bool USBOSDM2_getZoomEnable(int devNum, bool &enable);

Function: Get Video Zooming Status of an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter enable: True means zooming is enabled.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setZoomMode(int devNum, bool HandV);

Function: Set Video Zooming Mode for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter HandV: True (default) means zooming is horizontal and vertical, false means zooming is horizontal only.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setZoomBoundary(int devNum, bool b);

Function: Enable / Disable Video Zooming Area's Boundary for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter b: True means zooming area has boundary, false (default) means zooming area has no boundary.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getZoomBoundary(int devNum, bool &b);

Function: Get Video Zooming Area's Boundary Status for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter b: True means zooming area has boundary.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setZoomBoundaryColor(int devNum, unsigned char bc);

Function: Set Video Zooming Area's Boundary Colour for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter bc: 0 = Black (default), 1 = 25% Gray, 2 = 75% Gray, 3 = 100% White

Return: True for success.

USBOSDM2_API bool USBOSDM2_getZoomBoundaryColor(int devNum, unsigned char &bc);

Function: Get Video Zooming Area's Boundary Colour of an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter bc: 0 = Black (default), 1 = 25% Gray, 2 = 75% Gray, 3 = 100% White

Return: True for success.

USBOSDM2_API bool USBOSDM2_setZoomPoint(int devNum, unsigned short x, unsigned short y);

Function: Set Video Zooming Area's Starting Point for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter x: Zooming area's starting point X co-ordinate in pixel unit.

Input Parameter y: Zooming area's starting point Y co-ordinate in pixel unit.

Return: True for success.

Note: (x, y) must be within full size video frame range for PAL(720, 576) or NTSC(720, 480).

USBOSDM2_API bool USBOSDM2_getZoomPoint(int devNum, unsigned short &x, unsigned short &y);

Function: Get Video Zooming Area's Starting Point of an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter x: Zooming area's starting point X co-ordinate in pixel unit.

Output Parameter y: Zooming area's starting point Y co-ordinate in pixel unit.

Return: True for success.

---- Channel Video Colour Functions

USBOSDM2_API bool USBOSDM2_setChanBrightness(int devNum, int chanNum, int bri);

Function: Set One Channel's Video Brightness for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter bri: New brightness, valid value range is:

if channel's video source is from input 1~4, value is within [-128, 127] inclusive, default is 0;

if channel's video source is from input 5, value is within [0, 255] inclusive, default is 128.

Return: True for success.

Note: Channel video signal source is set / retrieved by **USBOSDM2_setChanVideoSrc/USBOSDM2_getChanVideoSrc**.

USBOSDM2_API bool USBOSDM2_getChanBrightness(int devNum, int chanNum, int &bri);

Function: Get One Channel's Video Brightness for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter bri: the brightness value.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setChanContrast(int devNum, int chanNum, int con);

Function: Set One Channel's Video Contrast for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter con: New contrast, valid value range is:

if channel's video source is from input 1~4, value is within [0, 255] inclusive, default is 64;

if channel's video source is from input 5, value is within [0, 255] inclusive, default is 128.

Return: True for success.

Note: Channel video signal source is set / retrieved by **USBOSDM2_setChanVideoSrc/USBOSDM2_getChanVideoSrc**.

USBOSDM2_API bool USBOSDM2_getChanContrast(int devNum, int chanNum, int &con);

Function: Get One Channel's Video Contrast for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter con: The contrast value.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setChanHue(int devNum, int chanNum, int hue);

Function: Set One Channel's Video Hue for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter hue: New hue, valid value range is within [0, 128] inclusive, default is 0;

Return: True for success.

USBOSDM2_API bool USBOSDM2_getChanHue(int devNum, int chanNum, int &hue);

Function: Get One Channel's Video Hue for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter hue: the hue value.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setChanSharpness(int devNum, int chanNum, int sha);

Function: Set One Channel's Video Sharpness for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter sha: New sharpness, valid value range is:

if channel's video source is from input 1~4, value is within [0, 15] inclusive, default is 1;

if channel's video source is from input 5, value is within [0, 3] inclusive, default is 0.

Return: True for success.

Note: Channel video signal source is set / retrieved by **USBOSDM2_setChanVideoSrc/USBOSDM2_getChanVideoSrc**.

USBOSDM2_API bool USBOSDM2_getChanSharpness(int devNum, int chanNum, int &sha);

Function: Get One Channel's Video Sharpness for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter sha: the sharpness value.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setChanUSaturation(int devNum, int chanNum, int usa);

Function: Set One Channel's Video Saturation for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.
Input Parameter usa: New saturation value within [0, 255] inclusive, default is 128;
if channel's video source is from input 1~4, usa is U Saturation,
if channel's video source is from input 5, usa is Saturation.

Return: True for success.

Note: When a channel's video signal source is from input 1 ~ 4 (all are RCA sockets), the video colour saturation has separate U Saturation and V Saturation to independently adjust the digitized YUV colour's U and V component. Video signal from source socket 5 (either RCA or SVideo socket) does not have separate U or V saturation. Channel video signal source is set / retrieved by **USBOSDM2_setChanVideoSrc/USBOSDM2_getChanVideoSrc**.

USBOSDM2_API bool USBOSDM2_getChanUSaturation(int devNum, int chanNum, int &usa);

Function: Get One Channel's Video Saturation for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter usa:

if channel's video source is from input 1~4, usa is U Saturation;

if channel's video source is from input 5, usa is Saturation.

Return: True for success.

Note: See note for **USBOSDM2_setChanUSaturation()**.

USBOSDM2_API bool USBOSDM2_setChanVSaturation(int devNum, int chanNum, int vsa);

Function: Set One Channel's Video V Saturation for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter vsa: New V Saturation, must be within [0, 255] inclusive, default is 128.

Return: True for success.

Note 1: See note for **USBOSDM2_setChanUSaturation()**.

Note 2: If this channel's video source is from input 5 (either RCA or SVideo), this function returns false.

USBOSDM2_API bool USBOSDM2_getChanVSaturation(int devNum, int chanNum, int &vsa);

Function: Get One Channel's Video V Saturation for an **OSD Device**.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter vsa: the V Saturation value.

Return: True for success.

Note 1: See note for **USBOSDM2_setChanUSaturation()**.

Note 2: If this channel's video source is from input 5 (either RCA or SVideo), this function returns false.

---- Channel Background Colour Functions

Each **OSD Device** has a Background Colour which can be either Blue (default) or Black. When an input video channel displays the background colour the entire channel is covered by the background colour regardless if the channel has input video signal or not. These functions control how the background colour is displayed.

USBOSDM2_API bool USBOSDM2_setBkColourMode(int devNum, unsigned char mode);

Function: Set OSD Device's Video Channels' Background Colour Mode.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter mode: mode = 0: manual mode (default) – a channel displays background colour only when that

channel's background colour is specifically enabled by calling

USBOSDM2_setEnableBkColour;

mode = 1: auto mode – a channel displays background colour only when the channel has no input video signal.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getBkColourMode(int devNum, unsigned char &mode);

Function: Get OSD Device's Video Channels' Background Colour Mode.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter mode: mode = 0: manual mode, mode = 1: auto mode.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setEnableBkColour(int devNum, int chanNum, bool enable);

Function: Enable/Disable a Video Channel's Background Colour.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Input Parameter enable: True for enabling the channel's background colour, false (default) for disabling.

Return: True for success.

Note: This is only effective when the OSD Device's background colour mode is manual.

USBOSDM2_API bool USBOSDM2_getEnableBkColour(int devNum, int chanNum, bool &enable);

Function: Get a Video Channel's Background Colour enable/disable status.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device channel number, must be between [0, 3] inclusive.

Output Parameter enable: True means the channel's background colour is enabled.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setBkColour(int devNum, unsigned char col);

Function: Set OSD Device's Video Channels' Background Colour.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter col: col = 0: Set background colour to Blue(default), col = 1: Set background colour to Black.

Return: True for success.

Note 1: OSD Device's 4 input video channels have the same background colour.

Note 2: A channel will show background colour when either of these happens:

- (1) the channel's background colour is enabled (see **USBOSDM2_setEnableBkColour**) and the OSD Device's background colour mode is manual (see **USBOSDM2_setBkColourMode**).
- (2) the channel has no input video signal and the OSD Device's background colour mode is auto (see **USBOSDM2_setBkColourMode**).

USBOSDM2_API bool USBOSDM2_getBkColour(int devNum, unsigned char &col);

Function: Get OSD Device's Video Channels' Background Colour.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter col: col = 0: background colour is Blue, col = 1: background colour is Black.

Return: True for success.

4.3.2 Audio Input Channel Functions

USBOSDM2_API bool USBOSDM2_initAudioChips(int devNum);

Function: Initialize audio hardware for an OSD Device: each OSD Device must call this function at least once for its audio input to work. The initialization process will also mute all audio channels.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setAudioChanType(int devNum, int chanNum, bool mic);

Function: Select audio input channel's signal source for an OSD Device.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device audio channel number, must be 0 or 2 for the first and third audio input channel.

Input Parameter mic: True for using Microphone socket as audio input source, false (default) for using Line-in socket.

Return: True for success.

Note 1: When parameter mic is true for chanNum 0, the Audio Input Socket 1 will be used,
when parameter mic is true for chanNum 2, the Audio Input Socket 2 will be used.

Note 2: The second and fourth audio channels (chanNum 1 and 3) cannot use this function since they can only use Line-in as their signal source.

USBOSDM2_API bool USBOSDM2_setAudioChanGain

(int devNum, int chanNum, unsigned char LorRorAll, unsigned short gain, bool clearRegFirst);

Function: Set audio channel's input gain.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device audio channel number, must be between 0 ~ 3 inclusive.

Input Parameter LorRorAll: 0=Left Channel, 1 = Right Channel, 2 = Both Left and Right Channels

Input Parameter gain: audio gain value, must be within [0, 70] inclusive.

Input Parameter clearRegFirst: If TRUE register clearing command is sent to audio chip before sending gain command.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setAudioChanMute

(int devNum, int chanNum, unsigned char LorRorAll, bool muteOn, bool clearRegFirst);

Function: Set audio channel's mute condition.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: OSD Device audio channel number, must be between 0 ~ 3 inclusive.

Input Parameter LorRorAll: 0=Left Channel, 1 = Right Channel, 2 = Both Left and Right Channels

Input Parameter muteOn: True to mute the channel, false to un-mute the channel.

Input Parameter clearRegFirst: If TRUE register clearing command is sent to audio chip before sending mute command.

Return: True for success.

Note: Channel 0~2 will always be muted/un-muted together, their left/right sub-channels are also muted/un-muted together.

4.3.3 Overlay Functions

These functions control USBOSDM2 Device's Bitmap Graphics and Single Box overlay operations, note all X/Y/Width/Height units are in 2-Pixel resolution.

---- Bitmap Graphics Overlay Functions

USBOSDM2_API bool USBOSDM2_setEnableOSD(int devNum, bool enable);

Function: Enable/Disable Bitmap Graphics Overlay for an OSD Device, will not Single Box Overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter enable: True to enable Bitmap Graphics overlay, false to disable overlay.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getEnableOSD(int devNum, bool &enable);

Function: Return the Bitmap Graphics Overlay enabling status for an OSD Device.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Output Parameter enable: True means the Bitmap Graphics overlay is enabled.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setOSDAAlpha(int devNum, unsigned char alpha);

Function: Set Bitmap Graphics overlay's alpha value.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter alpha: 0 = 50% alpha, 1 = 50% alpha, 2 = 75% alpha, 3 = 25% alpha.

Return: True for success.

Note 1: Higher alpha means the overlay has more visibility against its background video.

Note 2: Once an Alpha value is set, all subsequent Bitmap Graphics overlay colours that have alpha enabled will use the same alpha value until the alpha value is changed again.

USBOSDM2_API bool USBOSDM2_getOSDAAlpha(int devNum, unsigned char &alpha);

Function: Get Bitmap Graphics overlay's current alpha value.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter alpha: Current Bitmap Graphics overlay's alpha value.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setOSDBlink(int devNum, unsigned char blink);

Function: Set Bitmap Graphics overlay's Blinking Interval.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter blink: 0 = 0.25 second, 1 = 0.5 second, 2 = 1 second, 3 = 2 second.

Return: True for success.

Note 1: Higher blinking interval value means the blinking is slower.

Note 2: Once a Blinking value is set, all subsequent Bitmap Graphics overlay colours that have Blink enabled will use the same Blinking value until the Blinking value is changed again.

USBOSDM2_API bool USBOSDM2_getOSDBlink(int devNum, unsigned char &blink);

Function: Get Bitmap Graphics overlay's Blinking Interval.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter blink: Current Bitmap Graphics overlay's blinking value.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setOSDPage(int devNum, unsigned char page);

Function: Set OSD Display Page Number: after calling this function, all Bitmap Graphics overlay contents showing on USBOSDM2 Device Video surface will be from this page.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter page: The OSD Display Page number, must be between [0, 5] inclusive.

Return: True for success.

Note 1: **USDOSDM2** has 6 OSD Display Pages numbered from 0 to 5, each has 720X576 Bytes memory. At any time only one of these pages is being used as the current display page: only Bitmap Graphics overlay contents from this current display page will appear on the OSD Device's Video Surface. By default the page 0 is the current page. Calling this function can change the current display page number therefore will put the content of any display page onto the video surface as overlay.

Note 2: The **USBOSDM2 SDK**'s default "Overlay Setup" Dialog Window always uses OSD Display Page 0 as the current display page, and uses OSD Display Pages 1~4 to store the bitmap images of all downloaded fonts. It also uses Page 5 as some temporary bitmap storage (such as the "TestColour" operation in the bundled application software).

```

USBOSDM2_API bool USBOSDM2_drawOSDBitmap(
    int devNum, // USBOSDM2 Device number (from 0 up)
    unsigned char OPMODE, // Operation Mode: 1=Bitmap Write, 2=Bitmap Fill, 3=Bitmap Move
    unsigned short START_HPOS, // destination bitmap start X
    unsigned short START_VPOS, // destination bitmap start Y
    unsigned short END_HPOS, // destination bitmap end X
    unsigned short END_VPOS, // destination bitmap end Y
    unsigned short START_HSRC, // source bitmap start X position
    unsigned short START_VSRC, // source bitmap start Y position
    unsigned char SRCLOC, // source bitmap page for the bitmap move operations:
                          // 0=Scratch page, 1=Display page
    unsigned char DSTLOC, // destination bitmap page for the bitmap write/fill/move operations:
                          // 0=Scratch page, 1=Display page
    unsigned char FILL_COLOR, // Only used for OPMODE==2(Block Fill): 0~255
    unsigned char *data, // bitmap data: each byte represents a pixel's CLUT entry
    unsigned int dataLen, // length of data in bytes
    unsigned char page // OSD Display Page to write data to (0~5)
);

```

Function: Draw a bitmap on one OSD Display Page or Scratch Page.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter OPMODE: Operation Mode: 1=Bitmap Write, 2=Bitmap Fill, 3=Bitmap Move

Input Parameter START_HPOS: destination bitmap start X

Input Parameter START_VPOS: destination bitmap start Y

Input Parameter END_HPOS: destination bitmap end X, must be \geq START_HPOS

Input Parameter END_VPOS: destination bitmap end Y, must be \geq START_VPOS

Input Parameter START_HSRC: source bitmap start X position, only used for OPMODE==3(Bitmap Move),

Input Parameter START_VSRC: source bitmap start Y position, only used for OPMODE==3(Bitmap Move),

Input Parameter SRCLOC: source bitmap page for the bitmap move operations: 0=Scratch page, 1=Display page

Input Parameter DSTLOC: destination bitmap page for bitmap write/fill/move operations: 0=Scratch page, 1=Display page

Input Parameter FILL_COLOR: Only used for OPMODE==2(Bitmap Fill): 0~255

Input Parameter data: bitmap data, must be \geq $(\text{END_HPOS} - \text{START_HPOS} + 1) * (\text{END_VPOS} - \text{START_VPOS} + 1)$ bytes long, each byte represents a pixel's CLUT entry position

Input Parameter dataLen: length of data in bytes

Input Parameter page: The OSD Display Page number, must be between [0, 5] inclusive.

Return: True for success.

Note 1: As OSD Display memory, each **OSD Device** has 6 OSD Display Page and one Scratch Page, each of them has a size of 720X576 Bytes. By calling function **USBOSDM2_setOSDPage**, any OSD Display Page can be used as the current page to show its contents on the video surface of the **OSD Device**. The content of the Scratch Page cannot appear on the video surface; therefore the Scratch Page can be used as a temporary storage page. The Scratch Page can be used during "Bitmap Move" (OPMODE==3) operation as either the destination or source page.

Note 2: The "Bitmap Write" operation (OPMODE == 1) writes a bitmap stored in "data" onto the destination page.

The "Bitmap Fill" operation (OPMODE == 2) fills an area in the destination page with one single colour.

The "Bitmap Move" operation (OPMODE == 3) copies an area in the source page to an area in the destination page.

Note 3: The "data" and "dataLen" parameters are only used for Bitmap Write (OPMODE == 1) operation, the "data" parameter must point to an array of enough memory to hold the bitmap pixels: each byte in the "data" array represents a colour index into the OSD Device's CLUT table(see function **USBOSDM2_loadCLUT**).

Note 4: The FILL_COLOR is used only in "Bitmap Fill" operation to represent a colour index into CLUT. To erase an existing OSD area on the video surface, do a "Bitmap Fill" operation with FILLCOLOR == 255.

Note 5: The START_HSRC and START_VSRC parameters are only used for "Bitmap Move" (OPMODE == 3) operation to represent the starting position in the source bitmap.

Note 6: The SRCLOC parameter is only used in "Bitmap Move" (OPMODE == 3) operation to indicate either the Scratch Page or the Display Page.

Note 7: During a "Bitmap Move" (OPMODE == 3) operation, if values of SRCLOC and DSTLOC are the same, the move operation will happen in the same page. For example, if the current OSD Display is page 0 then the following function calls will copy the overlay content at the lower right quarter onto the upper left corner of the video surface:
for PAL Video: **USBOSDM2_drawOSDBitmap(0, 3, 0, 0, 359, 287, 360, 288, 1, 1, 0, NULL, 0, 0);**
for NTSC Video: **USBOSDM2_drawOSDBitmap(0, 3, 0, 0, 359, 239, 360, 240, 1, 1, 0, NULL, 0, 0).**

Note 8: To erase Overlay content of an area on any page, use value 255 for FILL_COLOR in a "Bitmap Fill" operation.

**USBOSDM2_API void USBOSDM2_eraseOSDContents(int devNum,
 unsigned short x, unsigned short width,
 unsigned short y, unsigned short height,
 unsigned char pageMin, unsigned char pageMax);**

Function: On an OSD Device, erase bitmap OSD contents within an area on some OSD Display Pages.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter x: the horizontal start position in pixels of the area to erase OSD contents.

Input Parameter width: the horizontal width in pixels of the area to erase OSD contents.

Input Parameter y: the vertical start position in pixels of the area to erase OSD contents.

Input Parameter height: the vertical height in pixels of the area to erase OSD contents.

Input Parameter pageMin: the first OSD memory Display Page number on which to erase OSD contents, within [0, 5]

Input Parameter pageMax: the last OSD memory Display Page number on which to erase OSD contents, within [0, 5].

Note 1: x + width must be < 721, y + height must be < 577, and pageMin must be <= pageMax.

Note 2: If an OSD timer created by **USBOSDM2_drawOSDTimer** is within the erasing area, the timer will only be erased once by this function, and on the next second the timer will be redrawn: therefore the timer will continue to display after calling this function. Use **USBOSDM2_deleteOSDTimer** to permanently erase an OSD timer.

USBOSDM2_API void USBOSDM2_eraseAllOSDContents(int devNum);

Function: On an OSD Device, erase all OSD contents on all OSD Display Pages, also erase OSD timer if it exists.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

**USBOSDM2_API bool USBOSDM2_drawOSDText(
 int devNum,
 HWND videoWin,
 int x,
 int y,
 char *text,
 unsigned char fR,
 unsigned char fG,
 unsigned char fB,
 unsigned char bR,
 unsigned char bG,
 unsigned char bB,
 unsigned char bkMode,
 int point,
 bool bold,
 bool italic,
 unsigned char alpha,
 unsigned char blink,
 char *typeFace);**

Function: Draw a text string on one OSD device's video surface at the OSD Display Page 0.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter videoWin: The handle of the owner window for this **OSD** Device's corresponding **M2B** device's video display, as indicated by the **wnd** parameter in function **USBOSDM2_drawVideo**, or the **ownerWnd** parameter in function **USBOSDM2_initDevs**. This parameter is used to calculate the font height from the supplied "**point**" parameter. If this is an invalid window handle then the SDK's default Video Window's handle will be used instead.

Input Parameter x: The overlay text string's starting horizontal position on video frame, in pixel unit.

Input Parameter y: The overlay text string's starting vertical position on video frame, in pixel unit.

Input Parameter text: The text string to be overlaid on video surface, NULL will make this function to do nothing.

Input Parameters fR, fG, fB: The RGB colour components for the foreground colour.

Input Parameters bR, bG, bB: The RGB colour components for the background colour.

Input Parameter bkMode: The background mode for displaying text, must be either 1 for Transparent, or 2 for Opaque.

Input Parameter point: The text font's point size.

Input Parameter bold: True means using Bold font.

Input Parameter italic: True means using Italic font.

Input Parameter alpha: 0=no alpha(the text is fully visible), 1=0.50 alpha, 2=0.75 alpha, 3=0.25 alpha.
Input Parameter blink: 0=no blink, 1=blink every 0.25sec., 2=blink every 0.5sec., 3=blink every 1 sec., 4= blink every 2 sec.
Input Parameter typeface: The font type face name. NULL string means “Times New Roman”.

Return: True for success.

```
USBOSDM2_API bool USBOSDM2_drawOSDTimer(  
    int devNum,  
    HWND videoWin,  
    int x,  
    int y,  
    unsigned char fR,  
    unsigned char fG,  
    unsigned char fB,  
    unsigned char bR,  
    unsigned char bG,  
    unsigned char bB,  
    unsigned char bkMode,  
    int point,  
    bool bold,  
    bool italic,  
    unsigned char alpha,  
    char *typeface,  
    bool displayDateAndTime);
```

Function: Draw a timer (updating time every second) on an OSD device’s video surface at the OSD Display Page 0.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Input Parameter videoWin: The handle of the owner window for this **OSD** Device’s corresponding **M2B** device’s video display, as indicated by the **wnd** parameter in function **USBOSDM2_drawVideo**, or the **ownerWnd** parameter in function **USBOSDM2_initDevs**. This parameter is used to calculate the font height from the supplied “**point**” parameter. If this is an invalid window handle then the SDK’s default Video Window’s handle will be used instead.
Input Parameter x: The timer string’s starting horizontal position on video frame, in pixel unit.
Input Parameter y: The timer string’s starting vertical position on video frame, in pixel unit.
Input Parameters fR, fG, fB: The RGB colour components for the foreground colour.
Input Parameters bR, bG, fB: The RGB colour components for the background colour.
Input Parameter bkMode: The background mode for displaying timer, must be either 1 for Transparent, or 2 for Opaque.
Input Parameter point: The font’s point size used to draw timer.
Input Parameter bold: True means using Bold font to draw timer.
Input Parameter italic: True means using Italic font to draw timer.
Input Parameter alpha: 0=no alpha(the text is fully visible), 1=0.50 alpha, 2=0.75 alpha, 3=0.25 alpha.
Input Parameter typeface: The font type face name. NULL string means “Times New Roman”.
Input Parameter displayDateAndTime: True means drawing date and time, false means drawing time only.

Return: True for success.

Note 1: Each device can have at most one timer displayed at any time. If a timer is already drawn this function returns false.
Note 2: Timer has no blinking capability (never blink).
Note 3: When a timer is successfully displayed, the bitmap of its 12 characters(0~9, -, :) will be stored at the upper left corner of the Display memory’s Scratch Page: do not use **USBOSDM2_drawOSDBitmap** function (OPMODE == 3) to move any bitmap to this corner otherwise the timer’s character bitmap will be modified resulting in wrong display of the timer.

```
USBOSDM2_API bool USBOSDM2_redrawOSDTimer(int devNum, int x, int y, bool displayDateAndTime);
```

Function: Redraw the existing OSD timer to a new location and / or different format (with or without date).

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Input Parameter x: The timer string’s starting horizontal position on video frame, in pixel unit.
Input Parameter y: The timer string’s starting vertical position on video frame, in pixel unit.
Input Parameter displayDateAndTime: True means drawing date and time, false means drawing time only.

Return: True for success.

Note: If no OSD timer exists this function returns false and does nothing. Use **USBOSDM2_isOSDTimerOn** to test timer.

USBOSDM2_API bool USBOSDM2_deleteOSDTimer (int devNum);

Function: Delete current OSD timer if it exists.

Return: True for success. If failure happens or no OSD timer was drawn previously returns false.

USBOSDM2_API bool USBOSDM2_isOSDTimerOn(int devNum);

Function: Return if an OSD timer exists.

**USBOSDM2_API bool USBOSDM2_loadCLUT(int devNum,
unsigned char startIndex,
unsigned char len,
unsigned char *RGB,
bool alpha,
bool blink);**

Function: Load CLUT (Colour Look Up Table) entries into USBOSDM2 OSD Device.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter startIndex: index of the first entry in CLUT the load the data to, must be between [0, 255] inclusive.

Input Parameter len: number of entries in the CLUT to load data, must be between [1, 255] inclusive.

Input Parameter RGB: An array of 3 elements per row, each row represents the RGB colour bytes for one pixel.

Input Parameter alpha: True for enabling Alpha for the loaded entries.

Input Parameter blink: True for enabling Blink for the loaded entries.

Return: True for success.

Note 1: At any time an **OSD Device** can only use maximum 251 different colours for all the overlay pixels on the current OSD Display Page. A 256-Entry Colour Look Up Table (CLUT) is used by the **OSD Device** to indicate which 251 colours out of the $256 * 256 * 256 == 16777216$ possible RGB combination colours will be used as the current display colours. Although the CLUT has 256 entries (from number 0 to 255), the number 1,2,3,4 and 255 entries will always indicate “transparent”, meaning any overlay pixel using these entry values will not display any colour – therefore these entries can be used to “erase” existing overlay colours.

Note 2: If Alpha or Blink is True, all “len” entries in the CLUT will use Alpha and/or Blink. The Alpha value used will be decided by calling **USBOSDM2_setOSDAAlpha**, while the Blink interval used will be decided by calling **USBOSDM2_setOSDBlink**.

Note 3: startIndex + len must be < 256.

USBOSDM2_API bool USBOSDM2_loadCLUTFromBMP(int devNum, char *fileName);

Function: Load a BMP File's Colour Palette as OSD Device's CLUT (Colour Look Up Table).

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter filename: A bitmap (.bmp) graphics file that is:

- (1) 8-bit colour,
- (2) indexed,
- (3) DIBSectioned.

Return: True for success.

Note: The number 1,2,3,4 and 255 entries in the BMP file's colour palette will be replaced with the closest matched colours in the palette since these entries in the CLUT will be transparent – i.e., any overlay pixel using the colours represented by entry number 1,2,3,4 or 255 of the BMP's colour palette will actually use some approximate colours in the palette.

**USBOSDM2_API bool USBOSDM2_loadOSDBitmapFile(int devNum,
unsigned short x,
unsigned short y,
unsigned char bkMode,
bool usePalette,
char *fileName);**

Function: Load a .BMP Graphics File as overlay onto OSD Device's video surface.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter x: horizontal position of the loaded graphics overlay on the video surface in pixel unit.

Input Parameter y: vertical position of the loaded graphics overlay on the video surface in pixel unit.

Input Parameter bkMode: background mode of the loaded graphics overlay: 1=TRANSPARENT, 2=OPAQUE.

Input Parameter usePalette: True will use the loaded graphics file's palette to replace current SDK CLUT.

Input Parameter filename: Null string or a .BMP graphics file that is:

- (1) 8-bit colour,
- (2) indexed,
- (3) DIBSectioned.

Return: True for success.

Note: When filename is NULL or non-existing file name this function will present a dialog asking for graphics file name.

---- Single Box Overlay Functions

Each **USBOSDM2** device has 4 "Single Box" overlay items that can be enabled/disabled, set position, colour, boundary, alpha etc. options unrelated to the "Bitmap Graphics" overlay options and the "Display Page" numbers described in the previous function calls. Once enabled, these "Single Box" overlay will also sit on top of the "Bitmap Graphics" overlay wherever the two types of overlays overlap, i.e., "Single Box" overlay has higher display priority than "Graphics Bitmap" overlay when they overlap.

Each Single Box has "Plane" (the inner area of the box) and "Boundary" (the surrounding edge of the box) that can be independently enabled and coloured.

A typical application of the Single Box overlay is to display a partially transparent box with text in front to highlight the text.

USBOSDM2_API bool USBOSDM2_setSBPlaneEnable(int devNum, int boxNum, bool enable);

Function: Enable/Disable (Show/Hide) a Single Box overlay's Plane (the box's inner body area).

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter enable: True for enabling the box's plane (showing it on video surface), False for disabling it.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getSBPlaneEnable(int devNum, int boxNum, bool &enable);

Function: Get Enable/Disable (Show/Hide) status of a Single Box overlay's inner body area.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Output Parameter enable: The returned Enable/Disable status.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setSBPlaneColour(int devNum, int boxNum, unsigned char colour);

Function: Select colour for a Single Box overlay's inner body area.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter colour: must be within [0, 15] inclusive:

- 0 White (75% Amplitude 100% Saturation) (default),
- 1 Yellow (75% Amplitude 100% Saturation),
- 2 Cyan (75 % Amplitude 100 Saturation),
- 3 Green (75% Amplitude 100% Saturation),
- 4 Magenta (75% Amplitude 100% Saturation),
- 5 Red (75% Amplitude 100% Saturation),
- 6 Blue (75% Amplitude 100% Saturation),
- 7 0% Black,
- 8 100% White,
- 9 50% Gray,
- 10 25% Gray,

- 11 Blue (75% Amplitude 75% Saturation),
- 12 Single Box Plane CLUT entry 0 colour, see function **USBOSDM2_setSBPlaneCLUT**,
- 13 Single Box Plane CLUT entry 1 colour, see function **USBOSDM2_setSBPlaneCLUT**,
- 14 Single Box Plane CLUT entry 2 colour, see function **USBOSDM2_setSBPlaneCLUT**,
- 15 Single Box Plane CLUT entry 3 colour, see function **USBOSDM2_setSBPlaneCLUT**.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getSBPlaneColour(int devNum, int boxNum, unsigned char &colour);

Function: Get the current colour of a Single Box overlay's inner body area.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Output Parameter colour: will be within [0, 15] as described in function **USBOSDM2_setSBPlaneColour**.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setSBPlaneCLUT

(int devNum, int CLUTIndex, unsigned char R, unsigned char G, unsigned char B);

Function: Set CLUT(Colour Look Up Table) entry value for Single Box overlay's inner body area.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter CLUTIndex: the CLUT entry number, between [0, 3] inclusive.

Input Parameter R: the Red component value of the RGB colour of the CLUT entry.

Input Parameter G: the Green component value of the RGB colour of the CLUT entry.

Input Parameter B: the Blue component value of the RGB colour of the CLUT entry.

Return: True for success.

Note 1: Each **USBOSDM2** has a 4-entry CLUT (Colour Look Up Table) for Single Box Plane's colour, each entry in the CLUT can be set to any RGB colour value, and each Single Box can use any one of these 4-entry's value as the colour of its Plane (inner body area of the box).

Note 2: Single Box Plane's CLUT is totally un-related to the CLUT used by the Bitmap Graphics overlay.

USBOSDM2_API bool USBOSDM2_getSBPlaneCLUT

(int devNum, int CLUTIndex, unsigned char &R, unsigned char &G, unsigned char &B);

Function: Get CLUT(Colour Look Up Table) entry value for Single Box overlay's inner body area.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter CLUTIndex: the CLUT entry number, between [0, 3] inclusive.

Output Parameter R: the Red component value of the RGB colour of the CLUT entry.

Output Parameter G: the Green component value of the RGB colour of the CLUT entry.

Output Parameter B: the Blue component value of the RGB colour of the CLUT entry.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setSBAlpha(int devNum, unsigned char alpha);

Function: Set display alpha value for Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter alpha: = 0 for 50% alpha
 = 1 for 50% alpha
 = 2 for 75% alpha
 = 3 for 25% alpha.

Return: True for success.

Note 1: When alpha is enabled, all Single Boxes will use the same alpha value.

Note 2: Larger alpha value means more visibility of the of Single Box overlay against the background video.

USBOSDM2_API bool USBOSDM2_getSBAlpha(int devNum, unsigned char &alpha);

Function: Get display alpha value for Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter alpha: = 0 for 50% alpha
= 1 for 50% alpha
= 2 for 75% alpha
= 3 for 25% alpha.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setSBAlphaEnable(int devNum, int boxNum, bool enable);

Function: Enable/disable display alpha for a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter enable: True for enabling display alpha for this Single Box overlay, default is false (disabling alpha).

Return: True for success.

Note: When alpha is disabled (default), the Single Box is fully visible against the background video.

USBOSDM2_API bool USBOSDM2_getSBAlphaEnable(int devNum, int boxNum, bool &enable);

Function: Get the enable/disable status of display alpha for a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Output Parameter enable: True for enabling display alpha for this Single Box overlay.

Return: True for success.

**USBOSDM2_API bool USBOSDM2_setSBPos(int devNum,
int boxNum,
unsigned short left,
unsigned short top,
unsigned short width,
unsigned short height);**

Function: Set display position and size for a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter left: X value of the upper left position in the video surface, in pixel unit.

Input Parameter top: Y value of the upper left position in the video surface, in pixel unit.

Input Parameter width: Width of the Single Box in the video surface, in pixel unit.

Input Parameter height: Height of the Single Box in the video surface, in pixel unit.

Return: True for success.

Note: For PAL video, position and size should not exceed video frame size 720 X 576 pixels.

For NTSC video, position and size should not exceed video frame size 720 X 480 pixels.

**USBOSDM2_API bool USBOSDM2_getSBPos(int devNum,
int boxNum,
unsigned short &left,
unsigned short &top,
unsigned short &width,
unsigned short &height);**

Function: Get the display position and size of a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Output Parameter left: X value of the upper left position in the video surface, in pixel unit.

Output Parameter top: Y value of the upper left position in the video surface, in pixel unit.
Output Parameter width: Width of the Single Box in the video surface, in pixel unit.
Output Parameter height: Height of the Single Box in the video surface, in pixel unit.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setSBBoundaryEnable(int devNum, int boxNum, bool enable);

Function: Enable/disable boundary for a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter enable: True for enabling boundary for the Single Box overlay, false (default) for disabling boundary.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getSBBoundaryEnable(int devNum, int boxNum, bool &enable);

Function: Get boundary enable/disable status for a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Output Parameter enable: True for enabling boundary for the Single Box overlay, false for disabling boundary.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setSBBoundaryColour(int devNum, int boxNum, unsigned char colour);

Function: Set boundary colour for a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Input Parameter colour: must be within [0, 3]: 0 = 0% white (default), 1 = 25% white, 2 = 50% white, 3 = 75% white.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getSBBoundaryColour(int devNum, int boxNum, unsigned char &colour);

Function: Get boundary colour of a Single Box overlay.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter boxNum: the single box number, between [0, 3] inclusive.

Output Parameter colour: 0 = 0% white, 1 = 25% white, 2 = 50% white, 3 = 75% white.

Return: True for success.

4.3.4 Generic OSD Device Functions

USBOSDM2_API bool USBOSDM2_colourBar(int devNum, bool cbon);

Function: Turn on/off OSD Device's colour bar output.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter cbon: true for turning on colour bar, false for turning off colour bar.

Return: True for success

USBOSDM2_API bool USBOSDM2_colourBarOn(int devNum);

Function: If OSD Device's colour bar is on return true, otherwise return false.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

USBOSDM2_API bool USBOSDM2_blackAndWhite(int devNum, bool on);

Function: Turn on/off OSD Device's video black and white output.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter on: true for turning the video output into black and white, false for turning off black and white output.

Return: True for success

USBOSDM2_API bool USBOSDM2_loadOSDDevInitVals(int devNum, char *filename);

Function: load default initialization values for one OSD Device from an initialization file.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter filename: the text file containing init values for OSD device

Return: True for success.

Note: The default initialization file for PAL is OSDValsPAL.ini, for NTSC is OSDValsNTSC.ini.

USBOSDM2_API bool USBOSDM2_resetOSDDevice(int devNum);

Function: Reset an OSD Device hardware and reload initialization file according to current input signal type.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Return: True for success.

Note: On success, this function automatically calls function **USBOSDM2_loadOSDDevInitVals()** with parameter “filename” == OSDValsPAL.ini, or OSDValsNTSC.ini.

USBOSDM2_API void USBOSDM2_disableOSDtoM2BVideoOut(int devNum, bool disable);

Function: Disable the video signal output channel from OSD Device to M2B Device.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter disable: True for disabling the video output so that the M2B device has no video input, false for enabling the video channel.

Note: By default the video signal output from OSD to M2B is enabled, calling this function with “disable”== TRUE will result in no video preview on PC’s screen and recorded MPEG video will contain total blackness.

USBOSDM2_API bool USBOSDM2_disableOSDChanVideoIn(int devNum, int chanNum, bool disable);

Function: Disable the video signal input on a channel of the OSD Device.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter chanNum: the video channel number, between [0, 3] inclusive.

Input Parameter disable: True for disabling the video input on this channel, False for enabling the video input.

Return: True for success.

Note: By default all 4 channels on each OSD Device are enabled for video signal input.

4.4 M2B Device Functions

These functions control the **M2B** devices through some underlining DirectShow filters.

4.4.1 M2B Video Input Functions

USBOSDM2_API bool USBOSDM2_setM2BVideoSrc(int devNum, int videoSrc);

Function: Set **M2B Device**’s input video source connection with the **OSD Device**’s video output.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter videoSrc: 0 for Composite Connection (default), 1 for SVideo Connection.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getM2BVideoSrc(int devNum, int &videoSrc);

Function: Get **M2B Device**’s current input video source connection with the **OSD Device**’s output.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter videoSrc: 0 for Composite Connection, 1 for SVideo Connection.

Return: True for success.

```
USBOSDM2_API bool USBOSDM2_getVideoProcAmpRange(  
    int devNum,  
    int property,  
    int &pMin,  
    int &pMax,  
    int &pSteppingDelta,  
    int &pDefault,  
    int &pCapsFlags);
```

Function: Get **M2B Device**'s input video colour property values' range.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter property: One of: 0(Brightness), 1(Contrast), 2(Hue), 3(Saturation), 4(Sharpness).

Output Parameter pMin: the minimum property value.

Output Parameter pMax: the maximum property value.

Output Parameter pSteppingDelta: the smallest increment by which the property can change.

Output Parameter pDefault: the default value of the property.

Output Parameter pCapsFlags: 0 = the video property is controlled manually, 1 = video property is controlled automatically.

Return: True for success.

Note 1: This function is a direct mapping of DirectShow function IAMVideoProcAmp::GetRange.

Note 2: Various colour properties and their range/default value are:

Brightness: [0, 255], default 114

Contrast: [0, 127], default 62

Hue: [-128, 127], default 0

Saturation: [0, 127], default 66

Sharpness: [0, 15], default 8

```
USBOSDM2_API bool USBOSDM2_setVideoProcAmp(int devNum, int property, int value, int flag);
```

Function: Set **M2B Device**'s input video colour property value.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter property: One of: 0(Brightness), 1(Contrast), 2(Hue), 3(Saturation), 4(Sharpness).

Input Parameter value: the new value of the property.

Input Parameter flag: 0 = the video property is controlled manually, 1 = video property is controlled automatically.

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMVideoProcAmp::Set.

```
USBOSDM2_API bool USBOSDM2_getVideoProcAmp(int devNum, int property, int &value, int &flag);
```

Function: Get **M2B Device**'s input video colour property value.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter property: One of: 0(Brightness), 1(Contrast), 2(Hue), 3(Saturation), 4(Sharpness).

Output Parameter value: the current value of the property.

Output Parameter flag: 0 = the video property is controlled manually, 1 = video property is controlled automatically.

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMVideoProcAmp::Get.

```
USBOSDM2_API bool USBOSDM2_setVCRHorizontalLocking(int devNum, bool lock);
```

Function: Set **M2B Device**'s VCR horizontal locking to relax signal standard matching to better maintain sync.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter lock: True for setting lock, false for not setting lock.

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMAnalogVideoDecoder::set_VCRHorizontalLocking.

USBOSDM2_API bool USBOSDM2_getVCRHorizontalLocking(int devNum, bool &lock);

Function: Get **M2B Device**'s VCR horizontal locking status.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Output Parameter lock: True for lock is on, false for lock is off.

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMAnalogVideoDecoder::get_VCRHorizontalLocking.

USBOSDM2_API bool USBOSDM2_get_HorizontalLocked(int devNum, bool &lock);

Function: Get **M2B Device**'s input signal horizontal sync locking status.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Output Parameter lock: True means the video input horizontal sync is locked (signal is available), false means no signal.

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMAnalogVideoDecoder::get_HorizontalLocked.

USBOSDM2_API bool USBOSDM2_get_NumberOfLines(int devNum, int &numberOfLines);

Function: Get **M2B Device**'s number of horizontal lines in input video signal.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Output Parameter numberOfLines: the number of horizontal lines in the video signal (525 for NTSC, 626 for PAL/SECAM).

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMAnalogVideoDecoder::get_NumberOfLines.

USBOSDM2_API bool USBOSDM2_get_TVFormat(int devNum, int &tvformat);

Function: Get **M2B Device**'s input video signal format.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Output Parameter tvformat: one of the followings:

```
AnalogVideo_None      = 0x00000000,  
AnalogVideo_NTSC_M    = 0x00000001,  
AnalogVideo_NTSC_M_J  = 0x00000002,  
AnalogVideo_NTSC_433  = 0x00000004,  
AnalogVideo_PAL_B     = 0x00000010,  
AnalogVideo_PAL_D     = 0x00000020,  
AnalogVideo_PAL_H     = 0x00000080,  
AnalogVideo_PAL_I     = 0x00000100,  
AnalogVideo_PAL_M     = 0x00000200,  
AnalogVideo_PAL_N     = 0x00000400,  
AnalogVideo_PAL_60    = 0x00000800,  
AnalogVideo_SECAM_B   = 0x00001000,  
AnalogVideo_SECAM_D   = 0x00002000,  
AnalogVideo_SECAM_G   = 0x00004000,  
AnalogVideo_SECAM_H   = 0x00008000,  
AnalogVideo_SECAM_K   = 0x00010000,  
AnalogVideo_SECAM_K1  = 0x00020000,  
AnalogVideo_SECAM_L   = 0x00040000,  
AnalogVideo_SECAM_L1  = 0x00080000,  
AnalogVideo_PAL_N_COMBO = 0x00100000.
```

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMAnalogVideoDecoder::get_TVFormat.

USBOSDM2_API bool USBOSDM2_set_TVFormat(int devNum, int tvformat);

Function: Set **M2B Device**'s input video signal format.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.
Input Parameter tvformat: Must be one of the tvformat values described in **USBOSDM2_get_TVFormat**.

Return: True for success.

Note: This function is a direct mapping of DirectShow function IAMAnalogVideoDecoder::put_TVFormat.

4.4.2 M2B Video Preview Functions

USBOSDM2_API void USBOSDM2_controlWinOn(bool on, bool topMost = true);

Function: Turn on or off the SDK's Control Window

Input Parameter on: True will show the USBOSDM2 Control Window, false will hide it.

Input Parameter topMost: If true(default) then the **USBOSDM2**'s Control Window is on top of all windows beneath it at overlapped areas.

Return: True for success.

Note: The default status of the Control Window is off (hiding).

USBOSDM2_API bool USBOSDM2_isControlWinOn(void);

Function: Return true if the SDK's Control Window is visible, else return false

**USBOSDM2_API void USBOSDM2_videoWinOn(bool on, char *m2bList = NULL,
HWND *parentWnd = NULL,
RECT *posRect = NULL,
bool topMost = true);**

Function: Turn on/off (show/hide) SDK's default "Video Window" and each individual M2B Device's video display.

Input Parameter on: If TRUE the SDK's "Video Window" (the default frame window holding individual M2B's video) will appear, otherwise it hides.

Input Parameter m2bList: This byte array's i'th member indicates if to turn on the i'th M2B's Video,
i = 0 ~ (total USBOSDM2 device number - 1). A non-zero value means turn on Video.

Input Parameter parentWnd: An array, the i'th member of the array indicates the parent window of the i'th M2B.

Input Parameter posRect: same meaning as in function **USBOSDM2_initDevs()**.

Input Parameter topMost: If true(default) the SDK's Video Window is on top of all windows beneath it at overlapped areas.

Note 1: If a member of parentWnd is NULL then the corresponding M2B device's video window has no parent window (parent window is desktop window).

If parentWnd itself is NULL then all M2B devices' video windows have no parent window.

Note 2: If m2bList is not NULL then all M2B devices' video will be turned on (appear on screen) or off (disappear from screen) depending on the values of each of m2bList's members: non-zero means on, zero means off.

If m2bList is NULL then the video windows' on/off status of all M2B devices will not change.

Note 3: By default the parent window (not the "parentWnd" array variable) of the SDK's default Video Window is NULL, so when mouse left button is held down inside it this window can be dragged around PC screen. Similarly when left mouse button double-clicked inside the video window it will toggle between full screen mode and normal window mode. However, when the parent window of the SDK default Video Window is explicitly set to non-NULL by the application software (see the VisualBasic Sample Code when the "Default Video Win" CheckBox is ticked), these left mouse single down + dragging and double-clicking behavior will not work anymore, since in this case the Video Window is a child window of some window created by the application software.

USBOSDM2_API bool USBOSDM2_isVideoWinOn(void);

Function: Returns if the SDK's Video Window is on or off PC's screen.

USBOSDM2_API void USBOSDM2_maximizeWindow(int devNum);

Function: If the USBOSDM2 default Video Window is visible, maximize it to occupy full screen (in full screen mode).
If the Video Window is not visible this function does nothing

Input Parameter devNum: three cases:

(1) if devNum is between [0, USBOSDM2_totalDevs() - 1] inclusive, maximize the video window to show only

USBOSDM2 device devNum's video

- (2) if devNum is -1, maximize the video window with its current content
 - (3) if devNum is -2, maximize the video window to show the **USBOSDM2** Device whose video area was been last clicked by left mouse button.
- If devNum is any other value, or if video window is already maximized, this function does nothing.

Note: This function is only effective when the parent of the SDK's default Video Window is NULL.

USBOSDM2_API bool USBOSDM2_isMaximized(int devNum);

Function: Return if a **USBOSDM2**'s Video Window is maximized (in full screen mode).

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Return: True means the device is maximized (in full screen mode).

USBOSDM2_API void USBOSDM2_restoreWindow(void);

Function: Restore the **USBOSDM2** video window to its previous status if it is currently maximized.

Note: This function is only effective when the parent of the SDK's default Video Window is NULL.

USBOSDM2_API void USBOSDM2_minimizeWindow(void);

Function: Minimize the **USBOSDM2** video and control window if they are currently not minimized.

Note: This function is only effective when the parent of the SDK's default Video Window is NULL.

USBOSDM2_API void USBOSDM2_drawVideo(int devNum, HWND wnd, int left, int top, int width, int height);

Function: draw a **M2B** Device's video within a rectangle (left, top, width, height) in the client area of window "wnd".

Input Parameter devNum: **USBOSDM2** device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive

Input Parameter wnd: Handle of the Window whose client area will be used to draw the video of this **M2B** device.

Input Parameter left, top, width, height: define the rectangle within the wnd's client area where video will be drawn, the upper left corner of the client area is always (left=0, top=0).

Note: This function can be used in a window's size changing function to redraw the video when the window size changes.

USBOSDM2_API void USBOSDM2_redrawVideo(void);

Function: Redraw SDK default video window's content for a **USBOSDM2** device.

Note: This function has no effect if the SDK default video window is not visible or not used to hold **M2B** device's video.

USBOSDM2_API bool USBOSDM2_pauseResumeVideo(int devNum, bool pause);

Function: Pause or resume-from-pausing the video display on PC's screen (will not affect video recording).

Input Parameter devNum: **USBOSDM2** device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive

Input Parameter pause: TRUE to pause video display on PC screen,

FALSE to resume video display on PC screen.

Return: TRUE for success.

Note: if "pause" is true but the video is already paused then this function returns false and no action will be taken, similarly if "pause" is false and the video is already displaying this function returns false without any effect.

USBOSDM2_API void USBOSDM2_disableVideo(int devNum, bool disable);

Function: Disable or enable the video and recording operation of a **M2B** device.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Input Parameter disable: TRUE to disable video display and recording,

FALSE to resume video display and recording.

4.4.3 M2B Audio Preview Functions

USBOSDM2_API bool USBOSDM2_mutePCSpeakers(int devNum, bool mute);

Function: mute or un-mute a USBOSD Device's audio speaker output on PC.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter mute: TRUE to mute this device's speaker output on PC,

FALSE to un-mute the PC speakers output.

Return: TRUE for success.

Note: Mute/Un-mute PC Audio Speaker output has no impact on **USBOSDM2** Device's audio input, nor any impact on recorded MPEG video file's audio volume.

USBOSDM2_API bool USBOSDM2_setPCSpeakersVolume(int devNum, int volume);

Function: Set USBOSD Device's audio speaker output volume on PC.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter volume: Must be between [-10000, 0] inclusive. Mute is -10000, full volume is 0.

Return: TRUE for success.

Note: This function has no impact on **USBOSDM2** Device's audio input, nor on recorded MPEG video file's audio volume.

USBOSDM2_API bool USBOSDM2_getPCSpeakersVolume(int devNum, int &volume);

Function: Get USBOSD Device's audio speaker output volume on PC.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter volume: Will be between [-10000, 0] inclusive. Mute is -10000, full volume is 0.

Return: TRUE for success.

USBOSDM2_API bool USBOSDM2_setPCSpeakersBalance(int devNum, int balance);

Function: Set USBOSD Device's audio speaker output balance on PC.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter balance: Must be between [-10000, +10000], 0 means left and right speakers have the same output.

A value of -10,000 indicates that the right speaker has been disabled and only the left speaker is receiving an audio signal. A value of 0 indicates that both speakers are receiving equivalent audio signals. A value of 10,000 indicates that the left speaker has been disabled and only the right speaker is receiving an audio signal.

Return: TRUE for success.

Note: Changing PC Audio Speaker output balance has no impact on **USBOSDM2** Device's audio input, nor any impact on recorded MPEG video file's audio.

USBOSDM2_API bool USBOSDM2_getPCSpeakersBalance(int devNum, int &balance);

Function: Get USBOSD Device's audio speaker output balance on PC.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter balance: Will be between [-10000, +10000], see **USBOSDM2_setPCSpeakersBalance**.

Return: TRUE for success.

4.4.4 MPEG Encoding Functions

USBOSDM2_API bool

USBOSDM2_setCallbackFuncC(int devNum, USBOSDM2_callback_get_streamC callbackC);

USBOSDM2_API bool

USBOSDM2_setCallbackFuncS(int devNum, USBOSDM2_callback_get_streamS callbackS);

Function: Set up user-defined call-back function to receive encoded MPEG data from USBOSDM2 hardware.

Input Parameter devNum: the OSD Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter callbackC: a pointer to integer function of type USBOSDM2_callback_get_streamC.

Input Parameter callbackS: a pointer to integer function of type USBOSDM2_callback_get_streamS.

Return: Both **USBOSDM2_setCallbackFuncC** and **USBOSDM2_setCallbackFuncS** return true on success.

Note 1: On a successful setup, these call-back functions will be called repeatedly by the **M2B** device each time a chunk of encoded MPEG data (approx. 130Kbytes) is available. If the application software processes these data it must do this quickly, i.e., the functions **callbackC** and **callbackS** must return quickly to its caller (the **USBOSDM2 SDK**) to avoid losing the next chunk of MPEG data which will arrive very soon.

Note 2: **USBOSDM2_callback_get_streamC** and **USBOSDM2_callback_get_streamS** are defined in **USBOSDM2.h** file as:
`typedef int (__cdecl * USBOSDM2_callback_get_streamC)(int devNum, unsigned char *buf, unsigned int length);`
`typedef int (__stdcall * USBOSDM2_callback_get_streamS)(int devNum, unsigned char *buf, unsigned int length);`

Both **USBOSDM2_callback_get_streamC** and **USBOSDM2_callback_get_streamS** accept the same type of parameters (**int devNum**, **unsigned char *buf**, **unsigned int length**), the only difference between them is **USBOSDM2_callback_get_streamC** is a "**_cdecl**" calling convention function pointer, where function caller clears stack after each function call --- this type of functions are used in C, C++, etc. programming languages, while **USBOSDM2_callback_get_streamS** is a "**_stdcall**" calling convention function pointer, where function itself clears stack after each function call --- this type of functions are used in VisualBasic etc. programming languages.

Note 3: When **USBOSDM2_setCallbackFuncC/USBOSDM2_setCallbackFuncS** is called with a pointer to a user-defined function, that user-defined function will be called by **USBOSDM2 SDK** each time **USBOSDM2 Device devNum** has received encoded MPEG data from the hardware MPEG encoder, the received data bytes are passed back in the "buf" parameter, the data's length in bytes is passed back in "length" parameter, the data generating **USBOSDM2 Device's** number is passed back in "devNum" parameter (between [0, **USBOSDM2_totalDevs()** - 1] inclusive).

Note 4: If **USBOSDM2_setCallbackFuncC/USBOSDM2_setCallbackFuncS** function is called with a NULL to the **callbackC/callbackS** parameter, the previously set-up callback will be cleared, which equals to the default case: if this function has never been called with a valid function pointer, the **USBOSDM2 SDK** will never call any user-defined function when it receives data from hardware MPEG encoder.

USBOSDM2_API bool USBOSDM2_setMPEGTVType(int devNum, int tv_type);

Function: Set MPEG video encoding TV Type of an **M2B** device.

Input Parameter **devNum**: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive
Input Parameter **tv_type**: 1 for PAL (default), 0 for NTSC.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGTVType(int devNum, int &tv_type);

Function: Get MPEG video encoding TV Type of an **M2B** device.

Input Parameter **devNum**: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive
Output Parameter **tv_type**: 1 for PAL , 0 for NTSC.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGVideoBitRate

(int devNum, unsigned int brate, unsigned int bratePeak, bool cbr);

Function: Set MPEG encoding bit rate for an **M2B** device.

Input Parameter **devNum**: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive
Input Parameter **brate**: MPEG Encoding bit rate in kilo bits per second (Kbps) unit. When **cbr** is false (for VBR), this is average bit rate. Valid bit rate values are from 1000 (1Mbps) to 25000 (25Mbps).

Input Parameter **bratePeak**: Maximum bit rate in Kbps unit for VBR encoding ("cbr" = false), max. 37500 (37.5Mbps).

Input Parameter **cbr**: True for CBR(Constant Bit Rate) encoding, False for VBR(Variable Bit Rate) encoding.

Return: True for success.

Note 1: The SDK's default setting is 8Mbps CBR: **brate** = 8000, **cbr** = True.

Note 2: When using VBR encoding ("cbr" = False), "brate" is the average bit rate, **bratePeak** is the maximum bit rate.
When using CBR encoding ("cbr" = True), "bratePeak" has no meaning.

USBOSDM2_API bool USBOSDM2_getMPEGVideoBitRate

(int devNum, unsigned int &brate, unsigned int &bratePeak, bool &cbr);

Function: Get MPEG encoding bit rate and CBR/VBR mode of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter brate: MPEG Encoding bit rate in kilo bits per second (Kbps) unit. When cbr is false (for VBR), this is average bit rate.
Output Parameter bratePeak: peak bit rate in kilo bytes unit, only meaningful when cbr is false (for VBR).
Output Parameter cbr: True for CBR(Constant Bit Rate) encoding, False for VBR(Variable Bit Rate) encoding.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGVideoResolution(int devNum, int video_resolution);

Function: Set MPEG encoding video frame size for an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter video_resolution: must be 0~3 inclusive:

For PAL Video:	0 = 720x576-Pixel
For NTSC Video:	0 = 720x480-Pixel
For PAL Video:	1 = 480x576-Pixel
For NTSC Video:	1 = 480x480-Pixel
For PAL Video:	2 = 352x576-Pixel
For NTSC Video:	2 = 352x480-Pixel
For PAL Video:	3 = 352x288-Pixel (MPEG1)
For NTSC Video:	3 = 352x240-Pixel (MPEG1)

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGVideoResolution(int devNum, int &hsize, int &vsize);

Function: Get MPEG encoding video frame size of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter hsize: the encoding horizontal frame size in pixel.
Output Parameter vsize: the encoding vertical frame size in pixel.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGVideoType(int devNum, int mpegtype);

Function: Set MPEG encoding output video stream type of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter mpegtype: must be of the following values to control MPEG encoding output type:

0 = MPEG2 Program Stream,
2 = MPEG1 System Stream,
10 = DVD Compliant MPEG2 Program Stream,
11 = VCD Compliant MPEG1 System Stream,
12 = SVCD Compliant MPEG2 Program Stream.

Return: True for success.

Note: When calling this function with parameter mpegtype==2(MPEG1) Or 11(VCD), make sure the encoding resolution is always set to 352X288 or 352X240 (using function **USBOSDM2_setMPEGVideoResolution**), because MPEG1 encoding requires video resolution to be 352X288 or 352X240. Encoding MPEG1 with other resolution can cause **M2B** device to crash.

USBOSDM2_API bool USBOSDM2_getMPEGVideoType(int devNum, int &mpegtype);

Function: Get current MPEG encoding output video stream type of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter mpegtype: current MPEG encoding output type as described in **USBOSDM2_setMPEGVideoType**.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGClosedGOP(int devNum, int cgop);

Function: Set MPEG encoding output video GOP(Group of Picture) type of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter cgop: GOP type: 0 = Open GOP, 1 = Closed GOP (default).

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGClosedGOP(int devNum, int &cgop);

Function: Get MPEG encoding output video GOP(Group of Picture) type of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter cgop: GOP type: 0 = Open GOP, 1 = Closed GOP (default).

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGIVTC(int devNum, int ivtc);

Function: Set MPEG encoding output video Inverse Telecine of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter ivtc: 0 = No Inverse Telecine(default), 1 = Yes.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGIVTC(int devNum, int &ivtc);

Function: Get MPEG encoding output video Inverse Telecine of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter ivtc: 0 = No Inverse Telecine, 1 = Yes.

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGAudioBitRate(int devNum, int audiodatarate);

Function: Set MPEG encoding Audio Data Bit Rate of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter audiodatarate: Must be one of the followings:
0x0A(192Kbps), 0x0B(224Kbps), 0x0C=(256Kbps:default), 0x0D(320Kbps), 0x0E(384Kbps).

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGAudioBitRate(int devNum, int &audiodatarate);

Function: Get MPEG encoding Audio Data Bit Rate of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter audiodatarate: See audiodatarate in **USBOSDM2_setMPEGAudioBitRate**

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGAudioSamplingRate(int devNum, int audiosamplingrate);

Function: Set MPEG encoding Audio Sampling Rate of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter audiosamplingrate: Must be one of: 2 (32KHz), 0(44.1KHz), 1 (48KHz: default).

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGAudioSamplingRate(int devNum, int &audiosamplingrate);

Function: Get MPEG encoding Audio Sampling Rate of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive

Output Parameter audiosamplingrate: Can be one of: 2 (32KHz), 0(44.1KHz), 1 (48KHz).

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGAudioOutputMode(int devNum, int aom);

Function: Set MPEG encoding Audio Output Mode of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter aom: Must be one of: 0(Stereo: default), 1(Joint), 2(Dual), 3(Mono).

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGAudioOutputMode(int devNum, int &aom);

Function: Get MPEG encoding Audio Output Mode of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter aom: Can be one of: 0(Stereo: default), 1(Joint), 2(Dual), 3(Mono).

Return: True for success.

USBOSDM2_API bool USBOSDM2_setMPEGAudioCRC(int devNum, int acrc);

Function: Set MPEG encoding Audio CRC of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter acrc: 0 for CRC Off (default), 1 for CRC On.

Return: True for success.

USBOSDM2_API bool USBOSDM2_getMPEGAudioCRC(int devNum, int &acrc);

Function: Get MPEG encoding Audio CRC of an **M2B** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Output Parameter acrc: 0 for CRC Off, 1 for CRC On.

Return: True for success.

USBOSDM2_API void USBOSDM2_setupEncodingParam (int devNum, HWND userWnd);

Function: Display a multi-tab dialog box for users to interactively set up all MPEG encoding parameters:



Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive
Input Parameter userWnd: must be the handle of a visible window to serve as the parent window of the displayed dialog box.

Note 1: On returning from calling this function(closing the multi-tab dialog window), **M2B** device will be reset.

Note 2: When the multi-tab dialog box is being displayed, **USBOSDM2 SDK** should never exit --- this means your

application software should reject any attempt to end **USBOSDM2 SDK** originated from the user when this function is being called (use function **USBOSDM2_dlgEncodingParamOn** to test this). One way to guarantee this is to disable exit menus/buttons from your application when this function is being called.

Note 3: The device name string (the text before “Device”) in the dialog box’s title can be changed by calling **USBOSDM2_setSoftwareName** function.

USBOSDM2_API bool USBOSDM2_dlgEncodingParamOn(int devNum);

Function: Test if the multi-tab dialog box invoked by **USBOSDM2_setupEncodingParam** is still being displayed.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Return: True if the **USBOSDM2_setupEncodingParam**-invoked dialog window has not ended (is still being displayed).

4.4.5 Video File Recording Functions

USBOSDM2_API void USBOSDM2_setRecordPathName(int devNum, char *pathname);

Function: Set recording file's path name excluding file name.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Input Parameter pathname: the new path name, preferably include full path from disk drive letter onwards, can have or not have trailing slash '\', NULL string will be ignored.

Note: “pathname” will not be checked to verify it is a valid path name string without illegal characters etc or if the path exists. Non-existing path name will NOT be created.

USBOSDM2_API void USBOSDM2_getRecordPathName(int devNum, char *pathname, int maxLen);

Function: Get recording path name excluding file name as a NULL-terminated string.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Output Parameter pathname: the current recording path name, including full path from disk drive letter onwards.

Input Parameter maxLen: must be > 2, indicating the maximum bytes that can be copied into “pathname”.

Note 1: Default recording path name is the folder where the **USBOSDM2.exe** program resides.

Note 2: If “pathname” is NULL or “maxLen” is < 3 this function does nothing.

USBOSDM2_API void USBOSDM2_setRecordFileName(int devNum, char *filename);

Function: Set recording file's file name excluding path name.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Input Parameter filename: the new file name without path, NULL string will be ignored. Length must be <= 128 bytes.

Note 1: filename will not be checked to verify it is a valid file name string without illegal characters etc.

Note 2: If “filename” is longer than 128 bytes it will be truncated to 128 bytes long.

Note 3: Recorded file name will always be appended with extension “.mpg”.

USBOSDM2_API void USBOSDM2_getRecordFileName(int devNum, char *filename, int maxLen);

Function: Get current recording file full name as a NULL terminated string.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive

Output Parameter filename: the current recording file name, including full path from disk drive letter onwards.

Input Parameter maxLen: must be > 0, indicating the maximum bytes that can be copied into “filename”.

Note : If “filename” is NULL or “maxLen” is < 1 this function does nothing.

USBOSDM2_API void USBOSDM2_setRecordFileNamingMethod(int devNum, int namingMethod);

Function: Set recording file naming method.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive

Input Parameter namingMethod: must be one of:

- 0(Default)= Prompt for file name Before Record,
- 1= Prompt for file name After Record,
- 2= Pre-name file using File Name Fields.

Note: Recorded file name will always be appended with .mpg.

USBOSDM2_API void USBOSDM2_setRecordTimer(int devNum, UINT timer);

Function: Set recording timer value.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive

Input Parameter timer: Value is in unit of minutes. Zero means cancel recording timer, this is the default setting.

USBOSDM2_API bool USBOSDM2_startRecord(int devNum, bool alsoStartStreaming);

Function: Start recording video to file on a USBOSDM2 device.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive

Input Parameter alsoStartStreaming: If true will also start streaming (if the device is not already streaming video).

Return: TRUE for success.

Note: If the device is already in recording mode this function does nothing and returns false.

USBOSDM2_API bool USBOSDM2_stopRecord(int devNum, bool alsoStopStreaming);

Function: Stop recording on a device.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter alsoStopStreaming: If true will also stop streaming (if the device is streaming video).

Return: TRUE for success.

Note: If the device is not in recording mode this function does nothing and returns false.

USBOSDM2_API unsigned __int64 USBOSDM2_getRecordLength(int devNum);

Function: Get the current MPEG recording file length in bytes.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Return: The recording file length in bytes. If the device is not recording this will be the last recorded file length.

USBOSDM2_API bool USBOSDM2_splitRecord(int devNum);

Function: If the USBOSDM2 device is recording split a new recording file and continue recording.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Return: True for success.

Note 1: If the device is not recording this function does nothing and returns false.

Note 2: New recording file name will be the same as the current recording file with a serial number appended to the end, see function **USBOSDM2_setSplitRecordSN**.

USBOSDM2_API void USBOSDM2_setSplitRecordSN(int devNum, unsigned int sn);

Function: Set recording file split serial number.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter sn: The next recording file split will append this number + 1 to the recording file name, default is 0.

Note: Each time a split file happens the split serial number will increase by one, until it reaches the reset number (when the reset number is not zero) then it will reset back to zero. See function **USBOSDM2_setSplitRecordSNReset** for the split number reset number.

USBOSDM2_API unsigned int USBOSDM2_getSplitRecordSN(int devNum);

Function: Get current recording file split serial number.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Output Parameter sn: The next recording file split will append this number + 1 to the recording file name.

Return: The next split number. Default number is 0.

USBOSDM2_API void USBOSDM2_setSplitRecordSNReset(int devNum, unsigned short resetSN);

Function: Set recording file split serial number's reset number.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter resetSN: If this value is non-zero, then when recording file split number reaches this value the split number will reset to 0. This reset number defaults to zero, meaning the recording file split number will always keep increasing by 1 each time a split happens until the recording stops.

USBOSDM2_API unsigned short USBOSDM2_getSplitRecordSNReset(int devNum);

Function: Get recording file split serial number's reset number.

Input Parameter devNum: Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Return: The current file split serial number's reset number. Zero means the split serial number will never reset to zero.

USBOSDM2_API bool USBOSDM2_deviceIsRecording(int devNum);

Function: Return if a **M2B** device is recording video to file.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive

Return: True means the device is recording.

USBOSDM2_API void USBOSDM2_pauseRecord(int devNum, bool pause);

Function: Pause or resume recording on an **M2B** device.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive

Input Parameter pause: True for pausing the recording if it's recording, false for resuming the recording if it's paused.

Note: If pause is true and the device is not recording or if pause is false and the device is not paused (inc. it's not recording at all) then this function does nothing and returns false.

USBOSDM2_API bool USBOSDM2_deviceIsPausedRecording(int devNum);

Function: Return if a **M2B** device is paused recording video to file.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Return: True means the device is paused recording, false means it is not paused which could mean not recording or recording but not paused: to differentiate between these two cases, use **USBOSDM2_deviceIsRecording()**.

4.4.6 Video Streaming Functions

USBOSDM2_API bool USBOSDM2_startStream(int devNum, bool writeFileToo);

Function: Start streaming on a **USBOSDM2** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter writeFileToo: True will also start recording (if the device was not already recording).

Return: TRUE for success.

Note 1: If the device is already in streaming mode this function does nothing and returns false.

Note 2: Video streaming is done in User Datagram Protocol (UDP) using the parameters set up in function **USBOSDM2_setStreamParams**.

**USBOSDM2_API bool USBOSDM2_setStreamParams
(int devNum, char *IPAddr, int TCPPort, bool multicast);**

Function: Setup streaming parameters for a **USBOSDM2** device when it is not streaming.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter IPAddr: Must be a legal IP address string such as 192.168.20.1 etc, default is 127.0.0.1.

Input Parameter TCPPort: Must be a valid port number, default is 5000.

Input Parameter multicast: True for multicast streaming (so "IPAddr" must be multicast address), false for un-cast (default).

Return: TRUE for success.

Note 1: If the **USBOSDM2** device is streaming this function does nothing and returns false.

Note 2: Multicast IP addresses are usually in the range 224.0.0.0 ~ 239.255.255.255.

USBOSDM2_API bool USBOSDM2_getStreamParams
(int devNum, char *IPAddr, int &TCPPort, bool &multicast);

Function: Get the current streaming parameters for a **USBOSDM2** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Output Parameter IPAddr: Must be long enough to hold the current IP address string such as 192.168.200.125 etc.

Output Parameter TCPPort: Current TCP/IP port number.

Output Parameter multicast: True for multicast streaming , false for un-cast streaming.

Return: TRUE for success.

USBOSDM2_API bool USBOSDM2_stopStream(int devNum);

Function: Stop streaming on a **USBOSDM2** device.

Input Parameter devNum: **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Return: TRUE for success.

Note: If the device is not in streaming mode this function does nothing and returns false

USBOSDM2_API bool USBOSDM2_deviceIsStreaming(int devNum);

Function: Return if a **M2B** device is streaming video to network.

Input Parameter devNum: **USBOSDM2** device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Return: True means the device is streaming.

Note: Streaming is independent of recording.

4.4.7 Still Image Grabbing Functions

USBOSDM2_API void USBOSDM2_captureOneImage(int devNum, int imageType, char *fileName);

Function: Capture the current video frame as image file.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Input Parameter imageType: Must be: 0=BMP, 1=GIF, 2=JPG, 3=PNG, 4=TIF.

Input Parameter filename: Must be a legal file name, preferably inc. full path (starting from the disk drive letter), also preferably without file name extension.

Note: On a successful image capture, file name extension will be automatically appended to “filename”
as .bmp, .gif, .jpg, .png, .tif.

4.4.8 Show Status Functions

These functions control how to display text strings in the PC Screen’s video window of the **USBOSDM2** device. These displayed text will not appear in the MPEG video data in recorded file and streamed video, nor in the TV Output sockets.

USBOSDM2_API bool USBOSDM2_showStatus(int devNum, char *text, int x, int y);

Function: Display text string on the PC’s video surface of an **OSDUSBM2** device if its status display is enabled (see function **USBOSDM2_enableStatus**).

The colour and font size of the displayed text is controlled by calling **USBOSDM2_setStatus**.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Input Parameter text: NULL-terminated string to display.

Input Parameter x: Horizontal start position of the displayed string inside the video window of devNum.

Input Parameter y: Vertical start position of the displayed string inside the video window of devNum.

Return: True for success.

Note 1: If the x/y position makes the displayed text exceeding current video frame edge the text will wrap around the edge.

Note 2: The current video frame dimension reflects the MPEG encoding’s video frame: e.g. DVD MPEG encoding will have 720X576 or 720X480 video frame dimension, while VCD MPEG encoding will have 352X288 or 352X240 video frame dimension.

USBOSDM2_API void USBOSDM2_enableStatus(int devNum, bool enable);

Function: Enable/disable a USBOSDM2 device's capability to display any status text inside its video window.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter enable: True (default) for enabling, false for disabling text display in the Video Window.

USBOSDM2_API void USBOSDM2_enableRecordStatus(int devNum, bool enable);

Function: Turn on/off a USBOSDM2 device's automatic display of recording status inside its video window.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter enable: True (default) for displaying, false for not displaying recording status.

Note: When enabled, recording status is automatically displayed by the SDK without any action from application software.

**USBOSDM2_API void USBOSDM2_setStatus(int devNum,
int X,
int Y,
int point,
unsigned char fR,
unsigned char fG,
unsigned char fB,
unsigned char bR,
unsigned char bG,
unsigned char bB,
int bkMode,
bool bold,
bool italic,
char *typeFace);**

Function: Set the position and font characteristics of the displayed status text inside the Video Window.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, USBOSDM2_totalDevs() - 1] inclusive.

Input Parameter X: The horizontal starting position in pixel unit of the status text, default is 0.

Input Parameter Y: The vertical starting position in pixel unit of the status text, default is 0.

Input Parameter point: The font point size of the status text, default is 18.

Input Parameter fR: The font foreground colour's Red component value of the status text, default is 255.

Input Parameter fG: The font foreground colour's Green component value of the status text, default is 255.

Input Parameter fB: The font foreground colour's Blue component value of the status text, default is 0.

Input Parameter bR: The font background colour's Red component value of the status text, default is 0.

Input Parameter bG: The font background colour's Green component value of the status text, default is 0.

Input Parameter bB: The font background colour's Blue component value of the status text, default is 0.

Input Parameter bkMode: The text display background mode, 1 (default) for Transparent, 2 for Opaque.

Input Parameter bold: True (default) will display the font in Bold.

Input Parameter italic: True will display the font in Italic, default is false.

Input Parameter typeFace: The Type Face of the display text font, maximum 32 byte long inc. the terminating NULL byte.
The default Type Face is "Times New Roman".

Note 1: The background colour is only used when bkMode is 2 (Opaque).

Note 2: The default foreground colour is yellow (RGB=255,255,0); the default background colour is black (RGB=0,0,0).

Note 4: If typeface has length > 32 it will be truncated to 32 bytes long including the terminating NULL (0) character.

Note 5: If typeface is NULL then the text font's Type Face will not change.

**USBOSDM2_API void USBOSDM2_getStatus(int devNum,
int &X,
int &Y,
int &point,
unsigned char &fR,
unsigned char &fG,
unsigned char &fB,
unsigned char &bR,
unsigned char &bG,
unsigned char &bB,
int &bkMode,**

bool &bold,
bool &italic,
char *typeface);

Function: Get the position and font characteristics of the displayed status text inside the Video Window.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Output Parameter X: The horizontal starting position in pixel unit of the status text.

Output Parameter Y: The vertical starting position in pixel unit of the status text.

Output Parameter point: The font point size of the status text.

Output Parameter fR: The font foreground colour's Red component value of the status text.

Output Parameter fG: The font foreground colour's Green component value of the status text.

Output Parameter fB: The font foreground colour's Blue component value of the status text.

Output Parameter bR: The font background colour's Red component value of the status text.

Output Parameter bG: The font background colour's Green component value of the status text.

Output Parameter bB: The font background colour's Blue component value of the status text.

Output Parameter bkMode: The text display background mode, 1 for Transparent, 2 for Opaque.

Output Parameter bold: True will display the font in Bold.

Output Parameter italic: True will display the font in Italic.

Output Parameter typeFace: Receiving the Type Face string of the display text font, must be at least 32-byte long.
This can be NULL in which case no type face string will be returned.

4.4.9 Generic M2B Device Functions

USBOSDM2_API bool USBOSDM2_resetM2BDevice(int devNum, bool redrawVideoWin);

Function: Reset **M2B** Device.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Input Parameter redrawVideoWin: If True, redraw the video content after reset.

Return: True for success.

Note: Resetting **M2B** device will interrupt recording, streaming operation if they are on.

USBOSDM2_API void USBOSDM2_disableM2BDevice(int devNum, bool disable);

Function: Disable or Enable **M2B** Device.

Input Parameter devNum: the **USBOSDM2** Device number, must be between [0, **USBOSDM2_totalDevs()** - 1] inclusive.

Input Parameter disable: If True, disable the **M2B** device, if False, enable **M2B** device.

Note 1: Once this function is called with **disable==TRUE**, all **M2B** functions will not work and should not be called until this function is called again with **disable==FALSE**.

Note 2: Disabling an **M2B** device will disconnect all DirectShow filters' pins in the underlining DirectShow graph used by the **M2B** device, therefore allowing separate DirectShow graph to be externally built using the same filters.

Note 3: Disabling **M2B** device will stop recording, streaming operation if they are on.

USBOSDM2_API void USBOSDM2_redrawVideoWin(void);

Function: Redraw the Video Window's content if the window is visible.

5. SDK Functions Calling Sequence

All **USBOSDM2** SDK functions can only be called after successfully calling the SDK start function **USBOSDM2_initDevs** and before calling the SDK end function **USBOSDM2_endDevs**, except the following functions that can be called anytime:

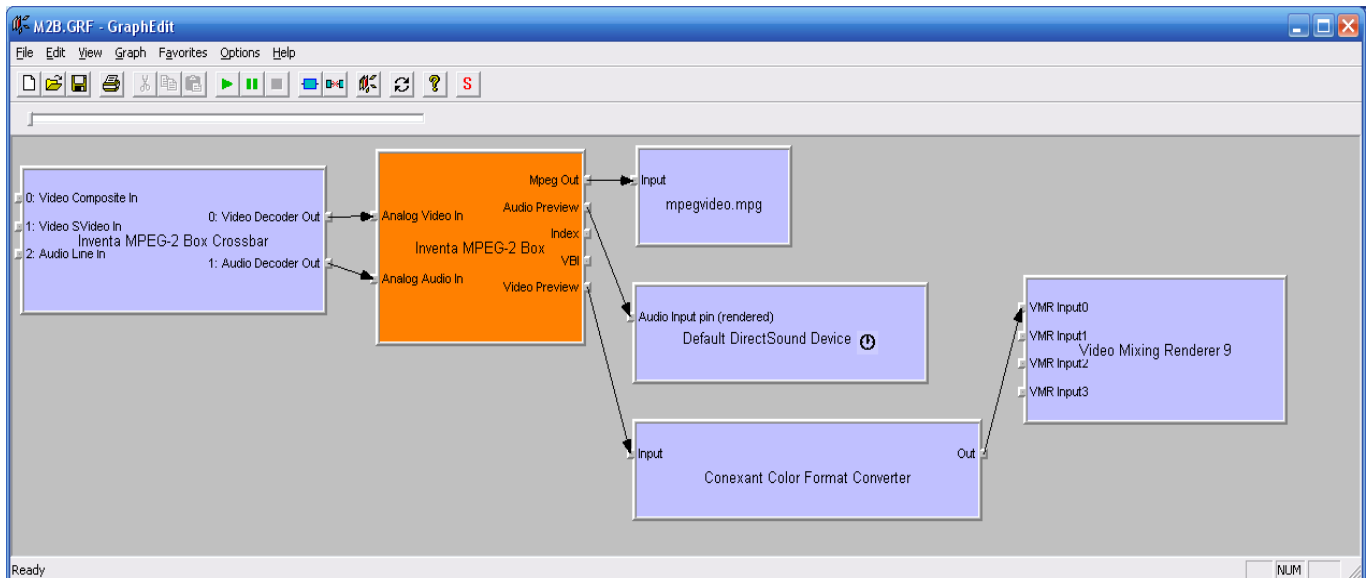
USBOSDM2_API void USBOSDM2_getSDKVer(char *ver);

USBOSDM2_API void USBOSDM2_setSoftwareName(char *name);

The **USBOSDM2** SDK requires function **USBOSDM2_initDevs()** to return > 0 to operate most of its functions, which means the PC has at least one **USBOSDM2** device properly connected and device drivers installed.

6. Using DirectShow Filters with the SDK

The **M2B Device** can be controlled directly by connecting Windows' DirectShow Filters without using the **USDOSDM2** SDK: to do this the function **USBOSDM2_disableM2BDevice(devNum, true)** must be called to disconnect the filters in the DirectShow graph built by the **USDOSDM2** SDK. The basic DirectShow Filter Graph to control **M2B** device is like this:



To operate the **M2B device** through DirectShow Filters separately like the above graph side by side with the **OSD device**, first call the **USBOSDM2 SDK**'s start function **USBOSDM2_initDevs** and any of the **OSD Device** functions, then call function **USBOSDM2_disableM2BDevice(devNum, true)** to disconnect the SDK's graph (therefore all **M2B device** functions of the SDK cannot be used any more). From there on a separate **M2B device** graph as shown in the previous picture can be freely constructed and connected. While the separate **M2B device** graph is operating, all **OSD device** functions of the **USBOSDM2 SDK** can still be called. When wishing to switch back to the SDK's **M2B device** functions, first stop and disconnect the separate DirectShow graph, then call **USBOSDM2_disableM2BDevice(devNum, false)** to reconnect the SDK's graph. From there on all **M2B Device** functions in the **USBOSDM2 SDK** can be used again.

7. SDK Installation & Running Environment

7.1 Install the SDK

From the **USBOSDM2 Setup CD**'s root folder, run **USBOSDM2SDKSetup.exe** will install all necessary libraries and filters onto the PC for using **USBOSDM2 SDK**. To uninstall the **SDK**, run **USBOSDM2SDKRemove.exe**.

7.2 Create Application with the SDK

On a PC with **USBOSDM2 SDK** installed as above, include the header file **USBOSDM2.h** and link the library **USBOSDM2.lib** with your source code will create an executable program that can dynamically call functions inside **USBOSDM2.dll** library. For programming languages that cannot directly use C-styled function declarations inside **USBOSDM2.h**, such as VisualBasic, Delphi etc., all needed **USBOSDM2 SDK** functions need to be individually declared, following the function names and parameter types listed in the **USBOSDM2.h** file.

To run the executable program linked with **USBOSDM2.lib**, the **USBOSDM2.dll** must be in a library searchable path, preferably in the same folder as the executable program itself; also in this folder with the **USBOSDM2.dll** file, the following files from the **USBOSDM2 SDK Setup CD**'s root folder must be present:

USBOSD.bix	--	The OSD Device firmware
USBOSDM2.bmp	--	Contains the default colour palette for Overlay
OSDValsPAL.ini	--	PAL format initialization file for OSD device
OSDValsNTSC.ini	--	NTSC format initialization file for OSD device
Input5Vals1.ini	--	Initialization file for OSD device for the 5 th Video Input

8. Sample Source Codes

Under the folder **Sample** folder on the **USBOSDM2 SDK Setup CD** there are Microsoft Visual C++ and VisualBasic sample source codes and MS Visual Studio projects that can be used to build application programs utilizing the **SDK**.

9. USBOSDM2 Hardware Specification

Host Interface: 2 X USB2.0 Type B Sockets
Power Supply: through USB Cables
Video Input: 5 X Composite (RCA), 1 X SVideo (4-Pin Mini-DIN)

Video Output (for Real-time Monitoring): 2 X Composite (RCA)
Audio Input: 4 X Line-in 3.5mm Stereo Mini Socket, 2 X Microphone 3.5mm Stereo Mini Jack
Audio Output: 2 X Line-out 3.5mm Stereo Mini Socket
Encoded Video Formats: MPEG2, MPEG1 MP@ML, Program Stream / System Stream
Constant Bit Rate (CBR) and Variable Bit Rate (VBR) Encoding
Video 4:2:2 to 4:2:0 Conversion
Video Inverse telecine (3:2 pulldown)
Video Encoding Frame Rates: 25 fps, 29.95 fps
Video Encoding Bit Rates: 1.00 Mbps ~ 25.00 Mbps
Video Encoding Resolution in Pixels: PAL: 352X288,480X576,720X576, NTSC: 352X240,480X480,720X480
Video Encoding Aspect Ratio: 4:3
Audio Encoding Format: MPEG1 Layer 2
Audio Sampling Rates: 32KHz, 44.1KHz, 48KHz
Audio Encoding Bit Rates: 192Kbps, 224Kbps, 256Kbps, 320Kbps, 384Kbps
Device Dimension: Top Width 138mm, Bottom Width 168mm, Depth 120mm, Height 40mm