USB Audio/Video Codec Model 2253

Windows Software Manual

Ver. 1.2.53 (May 2025)

SENSORAY | embedded electronics

Designed and manufactured in the U.S.A

SENSORAY | p. 503.684.8005 | email: info@SENSORAY.com | www.SENSORAY.com

7313 SW Tech Center Drive | Portland, OR 97203

Table of Contents

Operating System Support

The Windows SDK supports Windows XP (32 bit) and Windows 7 (32/64 bit). For Linux SDK see the Linux Software Manual.

Release Notes

Version 1.2.53 (May 2025)

- Fix “Mp4Remux stopped. Bad input!” error when recording MP4.

Version 1.2.52 (May 2025)

- Fix 16:9 aspect ratio for MPEG-4 video at PAL 720x576 and NTSC 720x480.

Version 1.2.51 (internal release)

Version 1.2.50 (August 2023)

- Add new deinterlacing functions S2253_GetDeinterlaceMode and S2253_SetDeinterlaceMode. Deinterlacing uses a hardware algorithm to smooth motion between two video fields that is visually more appealing than Interpolate mode, with a few limitations (see function description for details).

Version 1.2.49 (April 2023)

- Video renderer change for Windows 11 only. Default will be VMR9 if supported due to some issues with 64-bit VMR7 on Windows 11 (April 2023).

- EVR renderer previewtype added to S2253_SetPreviewType enumerations.

- Demo application allows changing the video renderer for test purposes (Default, VMR9, VMR7, EVR).

- Bug fix for FPS display under VMR9.

- Bug fix for switching between null and non-null windows with S2253_SetVideoWindow.

Version 1.2.48 (June 2022)

- Driver bug fix for use of the 2253 with some newer video cards. Please update if experiencing driver crashes when previewing the video stream.

Version 1.2.47 (April 2022)

- Fixed custom size of 768x576 for PAL video system. Fix includes driver, DLL, and demo application changes. Also adds 768x576 data-range enumeration for standalone DirectShow applications.

- There is a known issue with the demo application only. For the PAL video standard, the preview/record size of 720x480 should display as 720x576. If 720x480 is desired with the PAL video standard, please use the custom size dialog in the application.

Version 1.2.46 (February 2022)

- Driver fix for 32-bit processs/application on 64-bit Windows. There was a bug where the 32-bit proxy was not being registered the first time the device was installed on a PC. This would cause the 32-bit demo to fail loading (on 64-bit Windows only).

Version 1.2.45 (December 2021)

- WHQL Windows secure boot update for Win10/11.

Version 1.2.44 (November 2021)

- Driver update

Version 1.2.43 (November 2021)

- Driver update. Older driver cleanup fix. Win10 driver renamed to win_secureboot (usable for Win10 and Win11). Win secureboot driver not updated at this time.

Version 1.2.42 (August 2021)

- 2253 filter display name (aka friendly name) changeable via DIPswitch. Default is still Sensoray 2253 Capture A, or Sensoray 2253 Capture B, but other switch positions will append the switch position to the filter name. The appended DIP switch setting is not zero indexed.

- For easier identification of multiple boards (up to 4) in the system.

- Use S2253_GetDIPSwitch to retrieve the current DIP switch position.

- DIP switch 0: Sensoray 2253 Capture A, Sensoray 2253 Capture B
- DIP switch 1: Sensoray 2253 Capture A #2, Sensoray 2253 Capture B #2
- DIP switch 2: Sensoray 2253 Capture A #3, Sensoray 2253 Capture B #3
- DIP switch 3: Sensoray 2253 Capture A #4, Sensoray 2253 Capture B #4

Version 1.2.41 (July 2021)

- Fixes 2253 detection issue on latest versions of Windows. Update if 2253 not detected by the DLL/demo application.

Version 1.2.40 (June 2021)

- Update firmware to build 8937: comb filter re-enabled on PAL. Add three new level parameters: comb, luma, chroma filters.

Version 1.2.39 (June 2021)

- Update firmware to build 8936. Add Set/GetAudioChannels functions. Change StreamOverlay function to accept strmidx=3.

Version 1.2.38 (January 2021)

- Update firmware to build 8933: comb filter disabled on PAL.

Version 1.2.37 (November 2020)

- Audio preview fix for 2253 Player application.

Version 1.2.36 (April 2019)

- Firmware fix for PAL colorspace.

Version 1.2.35 (September 2018)

- S2253_GetLevel no longer caches the last set value. S2253_GetLevel will always send a USB request to the board to query the requested parameter.

Version 1.2.34 (July 2018)

- Firmware OSD text update

Version 1.2.33 (June 2018)

- Windows 10 version 1803 disables camera devices by default. Application (C++) launches camera privacy settings if access denied. Symptoms of this issue include a serial number of 0 on Windows 10.

Version 1.2.32

- Fix for Windows 7 32 bit driver.

Version 1.2.31

- Driver updates for Windows 10. Preview with callback bugfix.

Version 1.2.30

- Driver update for Windows 10 1607+ when secure boot is enabled.

Version 1.2.29 (September 2017)

- Updated firmware for styled overlay text and driver update for styled overlay text.

Version 1.2.28 (August 2017)

- Fix for styled overlay text lockup after long term operation. S2253_ResetBoard should not be necessary anymore.

- Functions S2253_SetAutoGain, S2253_SetAutoBrightness, S2253_GetAutoGain and S2253_GetAutoBrightness to turn off AGC refactored to S2253_SetLevel, S2253_GetLevel with S2253_LEVEL_AUTOGAIN and S2253_LEVEL_AUTOBRIGHTNESS parameters.

- Demo program “S2253 Player” fix for changing to PAL mode.

- Default driver turns AUTOGAIN and AUTOBRIGHTNESS off. Default SDK will turn AUTOGAIN and AUTOBRIGHTNESS on for backward compatibility with past SDKs.

Version 1.2.27 (July 2017)

- Memory leak fix for S2253_SetOsd with styled text.

- Adds functions S2253_SetAutoGain, S2253_SetAutoBrightness, S2253_GetAutoGain and S2253_GetAutoBrightness to turn off AGC.

Version 1.2.26 (May 2017)

- 2253P GPS view added to C# demo. Please see directory below for details. This directory is created after running setup.

- See C:\Program Files(x86)\Sensoray\2253\code_csharp for C# code.

Version 1.2.25 (April 2017)

- Fix low-latency mode and crop window parameters

- Update to C# demo for styled overlay text

- Added functions to get full Windows device path and instance for multiple boards if serial number not sufficient. S2253_GetFullDevicePath, S2253_GetFullDeviceInstance

Version 1.2.24 (June 2016)

- Fix MP4 mux

Version 1.2.23 (June 2016)

- Change in DirectShow filter PAL/NTSC setting for non-DLL use.

Version 1.2.22 (May 2016)

- Fix in the C demo: when starting the preview stream with PAL input is detected, and a settings config file is not present, would cause device video standard to switch to NTSC.

- Fix for overlay freeze: when overlays are updated frequently and preview stream is running over a long period of time.

Version 1.2.21 (March 2016)

- Fix for initial brightness, contrast, hue and saturation when board opened. Values will be saved between board open and close. Audio level meter added for audio preview under Audio Settings in demo app.

- S2253_StartAudioPreviewWithCallback,S2253_StartAudioCallback,S2253_StopAudioCallback,S2253_RegisterAudioCallback functions added.

Version 1.2.20 (February 2016)

- Fix for decode efficiency and raw decode bug fixes. Recommended update if using decode functionality. mid2253.dll, driver and SourceSray.dll, SourceSray_x64.dll updated.

Version 1.2.19 (February 2016)

- Added S2253P functions: ReadScaled, LoadScaled, SetScale, GetScale, which can be used for rotary encoder calibration.
- Firmware adds support to hardware-accelerated solid color rectangle overlays on Streams A & B for Styled Text, PNG and BMP images.

Version 1.2.18 (January 2016)

- Fix PCM audio in MPEG-2 Program Stream

Version 1.2.17 (January 2016)

- Fix MP4 fragmented stream and MP4 recording

Version 1.2.16 (December 2015)

- Bug fixes
- Fix for 953
- Use VMR9 by default for Windows 10

Version 1.2.15 (November 2015)

- Windows 10 fresh installation fix

Version 1.2.14 (November 2015)

- Add S2253_GetFPS function and use in preview status
- Implement mid2253_osd_bmp

Version 1.2.13 (October 2015)

- Driver signing update
- Letter Box preview option for VMR-7 and VMR-9 rendering
- Firmware update for overlays

Version 1.2.12 (March 2015)

- Preview drawing functions added
- Preview drawing with transparency

Version 1.2.11 (July 2014)

- Adding crop functions, and GPI pause trigger function.
- Adding freeze frame functions, and GPI freeze trigger function.
- Adding raw YUV frame playback.
- Adding fixed QP bitrate control.

Version 1.2.10 (February 2014)

- Support for Model 2253P: GPS, GPIO, and encoder API functions.

Version 1.2.9 (October 2013)

- Support for unicode and extended text (variable font) OSD.
- Support for pause/resume of recording.

Version 1.2.7 (April 2013)

- OSD text increased to 160 characters.
- Closed-caption data capture.

Version 1.2.1 (September 2011)

- MPEG decode feature added
- Ultra-low latency mode
- Video output overlays
- User data (for possible embedded GPS info or watermarks), GOP size, and IDR control

Version 1.1.1 (January 2011)

- Important re-distribution instructions added to manual
- Efficiency improvements and important fixes for multiple streams
- All SDK components (drivers, DLL, ActiveX) must be updated (see re-distribution chapter)

Version 1.1.0 (December 2010)

- Audio features added
- GPIO feature added
- Driver simplified (less devices in device manager)
- All SDK components must be updated

Version 1.0 (Summer 2010)

- Initial 2253 release.
- MPEG4, MJPEG, H264 Video encoder
- Raw video capture

Installation

The software may be distributed on a CD or downloaded from Sensoray's web site. If the file is downloaded, it will need to be unzipped into a folder on the local drive prior to connecting the 2253 to the USB port.

Setup is performed as follows.

1) Run `setup.exe` file.
2) Select "yes" when asked to pre-install drivers. Drivers are installed automatically. A local copy of the drivers is installed in the program files directory for the 2253 (typically, `C:\Program Files\Sensoray\2253\driver\x32` and `C:\Program Files\Sensoray\2253\driver\x64`).
3) Plug in the 2253 device.
4) On Windows XP, select "search for drivers automatically". On Windows 7 drivers should be loaded automatically.

Re-distribution

Re-distribution is when OEMs repackage the software as a customized application to their own customers. Re-distribution may be for initial installation on a customer's system or for a software update. If just using the demo apps, the above Installation section applies. Setup.exe will automatically update all SW and driver components. Setup.exe uses the NSIS installation system.

It is important to differentiate between drivers and DLLs. A driver is a separate component with an INF file and supporting files such as firmware. The DLL is a library of code. The DLL is not a driver and a driver is not a DLL.

If updating software, OEMs MUST install ALL software components. This includes all drivers, activeX components and DLL(s). An OEM's original application or .EXE should work with these new components without re-compilation unless otherwise indicated. Sensoray will not support customers who use different versions of the SW components. For example, upgrading only the DLL or firmware file without upgrading the driver is not supported. Using the new driver with the old DLL software will also not be supported by Sensoray support.

Because Sensoray does not design an OEM's end application, it is the OEM's sole responsibility to properly update software components with an appropriate SW

installer. Drivers should be updated with DPInst.exe or DIFx. Please refer to the MSDN documents online for more details on these tools if in doubt. Sensoray recommends users unfamiliar with driver installation to use Dpinst.exe. A copy of dpinst.exe is in the driver directory. This program will automatically update the drivers included in the same directory as dpinst.exe. The following is a list of components that MUST be always be updated anytime there is a new SW release. These files are installed after running setup.exe to the directory chosen (Program files\Sensoray\2253 in this case)

Driver:

Drivers in Program files\Sensoray\2253\driver\x32(for 32 bit Windows) or Program files\Sensoray\2253 \driver\x64 (for 64 bit Windows) must be installed using DPINST.exe or DIFx(for DIFx experts only). Copy the x32 or x64 (for 64 bit) directory to the target computer and run dpinst.exe from the same directory on the target computer. Good installation tools have provisions to detect if the target computer is 64 bit or not.

DLL(s) and ActiveX:

API\mid2253.dll

API\mp4remux.ax (register with regsvr32)

API\sraywrite.ax (register with regsvr32)

API\YUVxfm.dll (From GDCL http://www.gdcl.co.uk/downloads.htm register with regsvr32)

The mid2253.dll file being a DLL must be installed with care. Some customers may choose to install the DLL in the system32 directory (system-wide). If this is done, it must be ensured that no local copy of the DLL is present in the directory where the application (executable) runs. If a local copy is present in the application directory, that version will be used instead and it may be an older version. Sensoray's installer installs the DLL to the application directory, not to system32.

The files mp4remux.ax,YUVxfm.dll and sraywrite.ax should be registered with regsv32 after installation.

There are 2 drivers included with the 2253. Both drivers are equally important and both must be updated with redistribution. One of the driver's includes a firmware file. If the firmware file is updated, do not just overwrite the old firmware file in the windows system or driver directory as Windows can restore the old version without your control. Always use DIFx and DPInst to update both drivers and all files those

drivers contain. Never copy any driver files (s2253.fw or s2253av.sys) to the Windows driver's directory.

Basic operation

Video Capture Driver

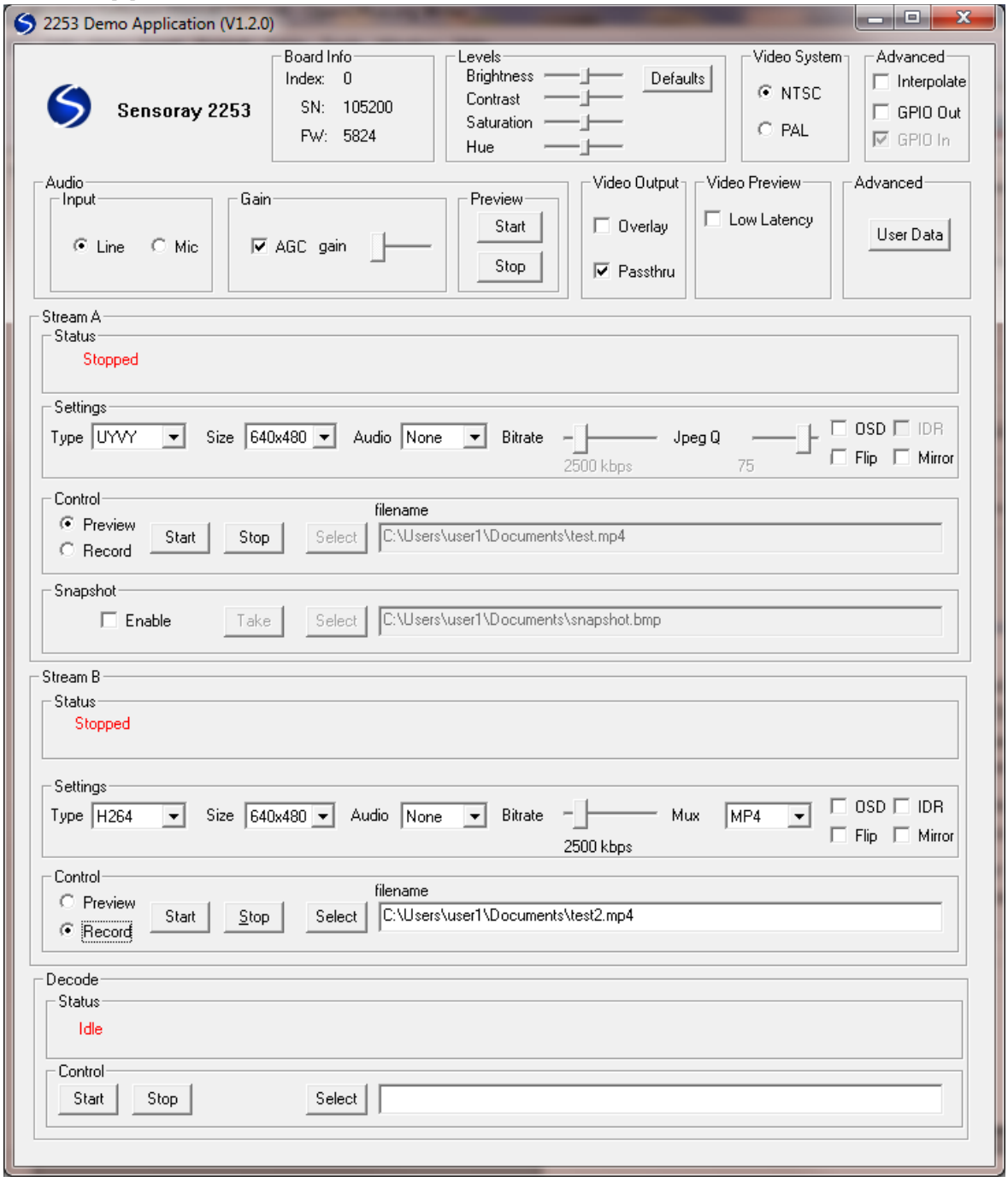
A 2253 device has 2 available streams. That allows a multitude of possible applications, like using an uncompressed stream for preview while recording the compressed stream. Another option is recording two streams with different resolutions and bitrates: a high quality stream for archiving, and a lower quality stream for low bandwidth streaming. Each stream shows up as a separate device in Windows Device Manager: Sensoray 2253 Multi-Codec Device (A) and Sensoray 2253 Multi-Codec Device (B). The DirectShow capture names for the streams are "Sensoray 2253 Capture A" and "Sensoray 2253 Capture B". If multiple 2253 devices are plugged in, then multiple devices will be present.

The two AVStream devices can be used with applications that support the DirectShow API or the Sensoray SDK. The Sensoray SDK is simply a wrapper around the DirectShow API to facilitate operation without knowledge of DirectShow programming. The DLL wrapper also allows easy porting to Visual Basic, C# and other programming environments. Video can be captured using uncompressed YUV422 (packed YUYV or UYVY) or YUV420SP (NV12 semi-planar, Y plane and interleaved CrCb plane), or encoded in compressed formats JPEG, MPEG4 or H.264 elementary streams. Both capture devices record video from a single source, and each capture device can be started, stopped, and configured independently. Some options cannot be configured independently, such as deinterlacing, brightness, hue, contrast, saturation.

GPIO Driver

GPIO support will be offered in later revisions of the SDK.

Demo application



1. Press Start button.
2. Select Programs.
3. Select Sensoray.

4. Select 2253.
5. Click on 2253 Demo.

The demo application by default is set up to preview the stream under Stream A and record to file in Stream B. The user may change the settings to record both streams (to separate files). Note that due to USB bandwidth constraints, previewing 2 streams in full color at full size is not supported.

Before starting any of the streams, it is necessary to set the video system, PAL or NTSC. Customers in North America will generally leave the setting as NTSC.

The next step is to select the video preview type. The options are UYVY, MJPG, or Y800. UYVY would be the most common setting. UYVY is an uncompressed color preview. MJPG is less efficient because the JPEGs must be decoded prior to display, but requires less USB bandwidth. MJPG preview is useful to see the effects of changing the JPEG Quality. Y800 is black and white preview and is the most efficient transfer of uncompressed video.

To get the lowest possible preview latency ensure the following parameters are configured (see Ultra-low Preview Latency below for details). Stream type:UYVY, Size :704x480 or 704x576, Flip: off (unchecked), Mirror: off (unchecked), Interpolate: off(unchecked) and Low latency: on (checked).

To start preview, press the Start button under StreamA->Control. A window will pop up with the stream.

The 2253 allows simultaneous recording while previewing uncompressed streams. Stream B in the demo app is set up to record by default to an H.264 encoded file. Press “Start” under Stream B to start recording to file. This can be done independently of the preview stream A.

DirectShow API

The driver supports the DirectShow API. DirectShow is well documented at MSDN (http://msdn.microsoft.com/en-us/default.aspx). The proxy file s2253proxy.h may be used to control the bitrate and some additional features of the 2253 through a COM interface. The operation of the 2253 proxy is shown in the DLL source code, which is also available with the full application source code. Please see the FAQ for an example of using the 2253 with a third party DirectShow program called VideoLan.

Ultra-low Latency Preview

The firmware on the 2253 has been optimized to provide as low a latency as possible for both compressed and uncompressed data. It should be noted, however, that lower latency is possible for uncompressed data under certain conditions. Ultra-low latency mode is enabled when the stream type is UYVY and the image is captured at native size (704x480 NTSC, 704x576 PAL) with no image flipping and no interpolation. This mode is automatically enabled when the parameters above are met. Please note that this mode is different from the S2253_LowLatencyPreview command described later in this manual. S2253_LowLatencyPreview only affects the Windows DirectShow host PC display latency, not the capture and USB transfer latency. For diagrams on how the ultra-low latency feature works, please visit http://www.sensoray.com/products/2253.htm.

SDK Reference

All API functions are declared using the following definition and the `__stdcall` calling convention:

```
#define MID2253_API extern "C" __declspec(dllimport)
```

For example,

```
MID2253_API int __stdcall S2253_Open (void);
```

All API functions return a value of type `int`, which is set to 0 on success, or a negative value if error (see `mid2253types.h` for error codes list).

Initialization/Cleanup/Enumeration Functions

S2253_Open

```
MID2253_API int __stdcall S2253_Open (int board_index);
```

`board_index`
Zero based index of a 2253 board (or -1 for all boards).

Must be called before any other API functions are called. If called with a -1 parameter, all 2253 boards in the system will be available after the call.

S2253_Close

```
MID2253_API int __stdcall S2253_Close (int board_index);
```

`board_index`
Zero based index of a 2253 board (or -1 for all boards).

Must be called before application terminates for proper clean-up of the SDK and SDK objects when SDK opened with S2253_Open.

S2253_GetNumDevices

```
MID2253_API int __stdcall S2253_GetNumDevices (
      int              *NumDevices);
```

Retrieves the number of devices in the system. Only valid after S2253_Open is called.

`NumDevices`
Address of a variable accepting the number of devices.

S2253_SetStreamWindow

```
MID2253_API int __stdcall S2253_SetStreamWindow (
      HWND              hwnd,
      int               devid,
      int               strmidx);
```

Optional function to preview in a predefined video window. If this function is not called or `hwnd` is `NULL`, the default pop-up window will display the video stream. If `hwnd` is not `NULL`, the window class should call `S2253_RepaintWindow` when a `WM_PAINT` message is received. An example MFC window class is shown in the Appendix.

`hwnd`
Window handle to render to. If NULL, default pop-up window will be used.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetStreamWindowPosition

```
MID2253_API int __stdcall S2253_SetStreamWindow (
      RECT              rcDst,
      int               devid,
      int               strmidx);
```

Optional function to set window position within an HWND. Default is to use entire entire. Does not apply if letter box format selected.

`rcDst`
Rectangle describing window position.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetVMRLetterBox

```
MID2253_API int __stdcall S2253_SetVMRLetterBox (
      BOOL              bLetterBox,
      int               devid,
      int               strmidx);
```

Optional function to maintain native video aspect ratio in the preview window by adding letter box to the video window. Default is off. Applies only to VMR-7 and VMR-9 preview render types.

`bLetterBox`
Whether letterbox format is on or not.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetVMRLetterBox

```
MID2253_API int __stdcall S2253_GetVMRLetterBox (
      BOOL              *bLetterBox,
      int               devid,
      int               strmidx);
```

Returns VMR letterbox setting.

`bLetterBox`
Whether letterbox format is on or not.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_RepaintWindow

```
MID2253_API int __stdcall S2253_RepaintWindow (
      HDC               hdc,
      int               devid,
      int               strmidx);
```

Used only if `S2253_SetStreamWindow` called with non-`NULL` `hwnd`. Call this function whenever the window in question must be repainted. For example, whenever it receives a `WM_PAINT` message.

`hdc`
Device context of the window in question. If NULL, default device context for the window handle will be used.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetSerialNumber

```
MID2253_API int __stdcall S2253_GetSerialNumber (
      unsigned int     *serial_number,
      int              devid);
```

Retrieves the serial number from the 2253. Each 2253 has a unique serial number.

`serial_number`
serial number of device.

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetDIPSwitch

```
MID2253_API int __stdcall S2253_GetDIPSwitch (
      unsigned int     *dip_switch,
      int              devid);
```

Retrieves the current DIP switch position from the 2253. The 2253 is shipped with default DIP switch setting of 0.

`dip_switch`
dip_switch position (0-3).

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetFullDevicePath

```
MID2253_API int __stdcall S2253_GetFullDevicePath (
      int              nWC
      WCHAR            *path,
      int              devid);
```

Returns the full USB Windows device path for the devid in question. This is an optional function only. Intended to be used if the serial number is not sufficient for a particular user's application. This function provides the USB port information inside a WCHAR string as well as other OS specific device information. It is up to the user to decode the path as desired for their application. Use S2263_GetFullDeviceInstance for a shorter version of the USB information.

`nWC`
Number of wide characters (WCHAR) in the path. Must be at least 260 or the function will fail.

`path`
full usb device path in WCHAR string format.

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetFullDeviceInstance

```
MID2253_API int __stdcall S2253_GetFullDeviceInstance (
      int              nWC
      WCHAR            *dev_instance,
      int              devid);
```

Same as S2253_GetFullDevicePath, but with the leading characters and the trailing GUIDs removed.

`nWC`
Number of wide characters (WCHAR) in the device instance. Must be at least 260 or the function will fail.

`dev_instance`
short version of the full usb device path in WCHAR string format.

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetVersions

```
MID2253_API int __stdcall S2253_GetVersions (
      MID2253_VERSIONS *version_info,
      int              devid);
```

Retrieves the version info for the SDK DLL and driver.

`version_info`
struct {
UINT32 dll_version;
UINT32 driver_version
UINT32 reserved[6]
};

`devid`
device id in the system (use 0 with a single board installed).

S2253_ResetBoard

```
MID2253_API int __stdcall S2253_ResetBoard (
      int              devid);
```

This function will reset the device. The device will disconnect from USB, reload the firmware, and start working again normally. The application must close and wait to reopen the board while it is resetting. For example:

```
S2253_ResetBoard(devid);
S2253_Close(devid);
do {
      Sleep(1000);
} while (S2253_Open(devid) != 0);
```

`devid`
device id in the system (use 0 with a single board installed).

GPIO Functions

S2253_GetGpioInput

```
MID2253_API int __stdcall S2253_GetGpioInput (
      int               *value,
      int               devid);
```

Retrieves current value of input GPIO.

`value`
pointer to returned GPIO value (1 if GPIO high, 0 if GPIO low).

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetGpioOutput

```
MID2253_API int __stdcall S2253_GetGpioOutput (
      int               *value,
      int               devid);
```

Retrieves current value of output GPIO.

`value`
pointer to returned GPIO value (1 if GPIO high, 0 if GPIO low).

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetGpioOutput

```
MID2253_API int __stdcall S2253_SetGpioOutput (
      int               value,
      int               devid);
```

Sets GPIO output on device. Please refer to hardware manual for connection details.

`value`
value of GPIO output to set (1 if GPIO high, 0 if GPIO low, 2 if GPIO output VSync signal).

`devid`
device id in the system (use 0 with a single board installed).

S2253_WaitGpioInput

```
MID2253_API int __stdcall S2253_WaitGpioInput (
      MID2253_GPIO_SIGNAL     signal,
      int               timeout,
      int               devid);
```

Waits for GPIO input. The input detection is edge triggered. This function will return right away if the input has already changed for the signal in question. For example, if the input goes from low to high before calling this function, S2253_WaitGpioInput will return right away for the signal MID2253_WAIT_HIGH. If S2253_WaitGpioInput is called again with MID2253_WAIT_HIGH, it will block until either the timeout or the input changes from low back to high to edge trigger the event. The driver sets an internal event on the change of signal. There is one event for the high transition and one event for the low transition. Calling this function will clear the event in question. Using MID2253_GPIO_CHANGE will wait for either event (or return right away if they have already been triggered since the last S2253_WaitGpioInput call).

`signal`
MID2253_GPIO_LOW, MID2253_GPIO_HIGH, MID2253_GPIO_CHANGE(high or low transition).

`timeout`
timeout in milliseconds. Use -1 for infinite.

`devid`
device id in the system (use 0 with a single board installed).

Status Functions

S2253_GetStatus

```
MID2253_API int __stdcall S2253_GetStatus (
      MID2253STATUS    *status,
```

```
      int               devid,
      int               strmidx);
```

Retrieves current status information (see `MID2253func.h` for `MID2253STATUS` type definition).

`status`
pointer to status variable.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetParam

```
MID2253_API int __stdcall S2253_GetParam (
      MID2253_PARAM    param,
      int              *val,
      int              devid,
      int              strmidx);
```

Retrieves informational parameters from the device.

`param`
parameter to retrieve. Currently only MID2253_PARAM_FIRMWARE (retrieve firmware version).

`val`
value of parameter retrieved.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetHVLock

```
MID2253_API int __stdcall S2253_GetHVLock (
      int              *lock,
      int              devid)
```

Returns video lock status.

`lock`
0 if unlocked or no signal, 1 if locked (HV locked).

`devid`

device id in the system (use 0 with a single board installed).

S2253_GetFPS

```
MID2253_API int __stdcall S2253_GetFPS (
      double            *pfps,
      int               devid,
      int               strmidx)
```

Returns the frames per second from the indicated preview stream.

`pfps`
pointer to the variable to receive the value in frames per second.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

Stream Control Functions

S2253_StartRecord

```
MID2253_API int __stdcall S2253_StartRecord (
      const void        *fileName,
      BOOL              bUnicode,
      int               devid,
      int               strmidx);
```

Starts recording to a file.

`fileName`
full path to the target file, no extension.

`bUnicode`
TRUE if filename is unicode.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_StartPreview

```
MID2253_API int __stdcall S2253_StartPreview (
      int               devid,
      int               strmidx);
```

Starts video stream and displays video in a pop-up window.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_EnableSnapshot

```
MID2253_API int __stdcall S2253_EnableSnapshot (
      BOOL              bOn,
      int               devid,
      int               strmidx);
```

Enables or disables snapshots (using `S2253_GetSample`) for preview or record streams. Disabling snapshots decreases CPU usage. Snapshots are enabled by default.

`bOn`
`TRUE` - enables `S2253_GetSample` function, `FALSE` - disables it.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_StartSnapshot

```
MID2253_API int __stdcall S2253_StartSnapshot (
      int               devid,
      int               strmidx);
```

Starts a snapshot stream. Device will stream and samples will be buffered until S2253_GetSample is called. Stream will not be previewed or recorded. Stop stream with S2253_StopStream. Snapshot stream is independent from preview or record stream. Snapshots can still be obtained with preview or record by using S2253_EnableSnapshot.

`devid`

device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetSample

```
MID2253_API int __stdcall S2253_GetSample (
      unsigned char     *data,
      unsigned int      inlen,
      unsigned int      *outlen,
      int               timeout,
      int               devid,
      int               strmidx);
```

Grabs sample data from a snapshot stream started with `S2253_StartSnapshot`, `S2253_StartPreview` or `S2253_StartRecord`. `S2253_GetSample` is used only for uncompressed (UYVY, Y800) or MJPEG streams.

`data`
a pointer to the sample data buffer.

`inlen`
length of data buffer.

`outlen`
pointer to the length of data copied.

`timeout`
how long to wait in milliseconds for next frame if data not available. Use 0 for non-blocking operation.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_StopStream

```
MID2253_API int __stdcall S2253_StopStream (
      int               devid,
      int               strmidx);
```

Stops streaming (record or video previewing, playing, previewing).

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_StartAudioPreview

```
MID2253_API int __stdcall S2253_StartAudioPreview (
      int               devid);
```

Starts independent audio stream for audio preview. The audio preview stream is independent of the video preview stream. Unlike the audio for recorded files, this stream is not synchronized to the video preview stream(s). Use S2253_StopAudioPreview to stop the audio preview.

`devid`
device id in the system (use 0 with a single board installed).

S2253_StopAudioPreview

```
MID2253_API int __stdcall S2253_StopAudioPreview (
      int               devid);
```

Stops the audio preview stream.

`devid`
device id in the system (use 0 with a single board installed).

S2253_StartDecode

```
MID2253_API int __stdcall S2253_StartDecode (
      const void        *fileName,
      BOOL              bUnicode,
      int               devid);
```

Starts decoding from a file. Decode is supported for H.264 and MPEG-4 stream types only. The 2253 hardware decoder is a closed decoder. As such, it will only decode streams recorded by the 2253. Like other hardware decoders, there is no guarantee it will decode other streams.

`fileName`
full path to the target file, no extension.

`bUnicode`
TRUE if filename is unicode.

devid
device id in the system (use 0 with a single board installed).

S2253_StartDecodeMem

```
MID2253_API int __stdcall S2253_StartDecode(
      int               devid);
```

Starts decoding from memory. Decode is supported for H.264 and MPEG-4 stream types only using S2253_DecodeData. Decode is supported for UYVY and Y8 streams using S2253_DecodeDataRaw (see also S2253_ForceDecodeFormat).

devid
device id in the system (use 0 with a single board installed).

S2253_DecodeData

```
MID2253_API int __stdcall S2253_DecodeData(
      unsigned char     *data,
      int               len,
      int               devid);
```

Sends data to the board, to be decoded and output on the physical Composite output. This command only supports playback of H.264 or MPEG-4 stream types.

data
pointer to data to decode

len
length of data.

devid
device id in the system (use 0 with a single board installed).

S2253_DecodeDataRaw

```
MID2253_API int __stdcall S2253_DecodeDataRaw(
      unsigned char     *data,
      int               len,
      int               devid);
```

Sends YUV data to the board, to be output on the physical Composite output. This command only supports playback of UYVY or Y8 data. After calling S2253_StartDecodeMem, call S2253_ForceDecodeFormat to set the YUV format and the image size.

data
pointer to data to decode

len

length of data.

`devid`
device id in the system (use 0 with a single board installed).

S2253_ForceDecodeFormat

```
MID2253_API int __stdcall S2253_ForceDecodeFormat(
      MID2253_DECODETYPE format,
      int                w,
      int                h,
      int                devid);
```

Sets the decode format if using YUV where auto-detectection of the type is not supported. Please note: If decoding compressed data, this function is not needed.

`format`
S2253_OUTFORMAT_UYVY or S2253_OUTFORMAT_Y8

`w`
width of image

`h`
height of image

`devid`
device id in the system (use 0 with a single board installed).

S2253_StopDecode

```
MID2253_API int __stdcall S2253_StopDecode (
      int                devid);
```

Stops decoding from a file.

`fileName`
full path to the target file, no extension.

`bUnicode`
TRUE if filename is unicode.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetNotify

```
MID2253_API int __stdcall S2253_SetNotify (
      HWND               hNotifyApp,
      UINT               mNotifyMsg,
      int                devid);
```

Sets a notify message, mNotifyMsg, that is sent to the application, hNotifyApp, when either decoding is finished or the device is unplugged. For decoding, this message, if registered using this function, is sent if the end of file is reached before S2253_StopDecode is called. The demo application has an example of how this function is used. Please note that S2253_StopDecode should be called after the file has finished decoding even if this message is received. Use S2253_TestDecodeDone and,or S2253_TestDeviceRemoval to check for device removal or decode finished in conjuncion with the notification message. A complete example is shown in the demo application.

`hNotifyApp`
handle to application to notify

`mNotifyMsg`
integer value of message

`devid`
device id in the system (use 0 with a single board installed).

S2253_TestDeviceRemoval

```
MID2253_API int __stdcall S2253_TestDeviceRemoval (
      int               devid);
```

To be used in conjuction with the notification message registered with S2253_SetNotify. Returns non-zero value if device removed.

`devid`
device id in the system (use 0 with a single board installed).

S2253_TestDecodeDone

```
MID2253_API int __stdcall S2253_TestDeviceRemoval (
      int               devid);
```

To be used in conjuction with the notification message registered with S2253_SetNotify. Returns non-zero value if decoding of a file is finished.

`devid`
device id in the system (use 0 with a single board installed).

S2253_StartCallback

```
MID2253_API int __stdcall S2253_StartCallback (
      int               devid,
      int               strmidx);
```

Start callback allows the user to capture data to a callback function that was previously registered with S2253_RegisterCallback. Care must be taken to minimize time spent in the callback routine otherwise the buffers used by DirectShow (the Windows capture API) will overflow. See S2253_RegisterCallback for details about the callback routine. Use S2235_StopStream to stop streaming to the callback function.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_RegisterCallback

```
MID2253_API int __stdcall S2253_RegisterCallback (
      cbfunc_t          callback,
      int               devid,
      int               strmidx);
```

Registers a callback. Care must be taken to minimize time spent in the callback routine otherwise the buffers used by DirectShow (the Windows capture API) will overflow.

`callback`
callback function to use. Callback function should be defined as follows: “int callback_name(BYTE *data, long size, int devid, int strmidx).

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetCallbackTimestamp

```
MID2253_API int __stdcall S2253_GetCallbackTimestamp (
      REFERENCE_TIME    *tstamp,
      REFERENCE_TIME    *now,
      int               devid,
      int               strmidx);
```

Get the internally stored timestamp for the current frame during the callback. The tstamp and now parameters receive the time at which the frame was captured, and the current time, respectively, counting from zero at the time when the stream was started. To calculate the latency, subtract tstamp from now. REFERENCE_TIME is

in units of 100ns.

`tstamp`
receives the time that the current frame was captured.

`now`
receives the current time of the stream.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_StartAudioCallback

```
MID2253_API int __stdcall S2253_StartAudioCallback (
      int               devid);
```

Start callback allows the user to capture audio preview data only to a callback function that was previously registered with S2253_RegisterAudioCallback. Care must be taken to minimize time spent in the callback routine otherwise the buffers used by DirectShow (the Windows capture API) will overflow. See S2253_RegisterAudioCallback for details about the callback routine. Use S2235_StopAudioCallback to stop streaming to the callback function.

`devid`
device id in the system (use 0 with a single board installed).

S2253_StopAudioCallback

```
MID2253_API int __stdcall S2253_StopAudioCallback (
      int               devid);
```

Stop streaming to audio callback.

`devid`
device id in the system (use 0 with a single board installed).

S2253_RegisterAudioCallback

```
MID2253_API int __stdcall S2253_RegisterAudioCallback (
      cbfunc_t          callback,
      int               devid,
      int               strmidx);
```

Registers an audio callback. Care must be taken to minimize time spent in the callback routine otherwise the buffers used by DirectShow (the Windows capture API) will overflow.

`callback`
callback function to use. Callback function should be defined as follows: “int callback_name(BYTE *data, long size, int devid, int strmidx).

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_StartPreviewCallback

```
MID2253_API int __stdcall S2253_StartPreviewCallback (
      int               devid,
      int               strmidx);
```

Start preview and callback, using callback functon registered by S2253_RegisterCallback. This function allows preview window to be used and also get preview image data in callback function. The same restrictions apply as with S2253_StartCallback.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_PauseStream

```
MID2253_API int __stdcall S2253_PauseStream (
      int               devid,
      int               strmidx);
```

Pause the recording or playback stream. Resume with S2253_ResumeStream. While paused, no video or audio frames are sent, effectively cutting the stream for the duration of the pause. See also: S2253_FreezeInput.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0, 1 or 2).

S2253_ResumeStream

```
MID2253_API int __stdcall S2253_ResumeStream (
      int               devid,
```

```
    int             strmidx);
```

Resumes a paused recording or playback stream.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0, 1 or 2).

S2253_PauseTrigger

```
MID2253_API int __stdcall S2253_PauseTrigger (
    int             devid,
    int             strmidx,
    S2253_PAUSE_MODE  mode);
```

This function configures a stream for pausing based on GPI event. The stream can be paused when one of the following events occur: rising edge, falling edge, level high, level low. If the high/low event condition is met when the stream is started, the stream will be paused and no frames delivered until the condition changes.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0, 1 or 2).

`mode`
S2253_PAUSE_MODE_DISABLE: stream will not pause
S2253_PAUSE_MODE_RISING_EDGE
S2253_PAUSE_MODE_FALLING_EDGE
S2253_PAUSE_MODE_LEVEL_HIGH
S2253_PAUSE_MODE_LEVEL_LOW

S2253_FreezeInput

```
MID2253_API int __stdcall S2253_FreezeInput (
    int             devid,
    bool            enable);
```

This function enables the video input to freeze a frame, meaning the same video frame is sent continuously to the recording and preview. This is different than pause; pause will stop sending video frames, cutting the duration from the recording. The freeze causes the recording itself to appear paused for the duration.

While freeze is enabled, the video can be cropped, mirrored or flipped, overlays may be updated, and audio will continue normally. Freeze affects both video streams.

`devid`
device id in the system (use 0 with a single board installed).

`enable`
boolean value to indicate true=freeze, false=resume

S2253_FreezeTrigger

```
MID2253_API int __stdcall S2253_FreezeTrigger (
      int               devid,
      S2253_PAUSE_MODE  mode);
```

This function configures video input freeze based on GPI event. The video input can freeze when one of the following events occur: rising edge, falling edge, level high, level low.

`devid`
device id in the system (use 0 with a single board installed).

`mode`
S2253_PAUSE_MODE_DISABLE: stream will not freeze
S2253_PAUSE_MODE_RISING_EDGE
S2253_PAUSE_MODE_FALLING_EDGE
S2253_PAUSE_MODE_LEVEL_HIGH
S2253_PAUSE_MODE_LEVEL_LOW

S2253_SetPreviewType

```
MID2253_API int __stdcall S2253_SetPreviewType (
      S2253_PREVIEWTYPE type,
      int               devid,
      int               strmid);
```

Windows has different video renderers available for preview display. This function controls which renderer is used. If this function is not called, the default renderer will be used, which is usually dependent on the Windows version and installed video drivers. If default is selected, Windows 11 will select VMR9 over VMR7 due to recent issues with VMR7 in 64-bit mode. Please note that MID2253_PREVIEWTYPE_EVR does not support operation with a null video handle (pop-up window). If using MID2253_PREVIEWTYPE_EVR, specify a handle with S2253_SetStreamWindow. Please note that MID2253_PREVIEWTYPE_LEGACY (which is not recommended) does not support a non-null video handle.

`type`

MID2253_PREVIEWTYPE_DEFAULT
MID2253_PREVIEWTYPE_VMR9
MID2253_PREVIEWTYPE_VMR7
MID2253_PREVIEWTYPE_LEGACY
MID2253_PREVIEWTYPE_EVR

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_DrawBitmap

```
MID2253_API int __stdcall S2253_DrawBitmap (
      HDC               hdcBMP,
      RECT              *src,
      NORMALIZEDRECT    *dst,
      float             alpha,
      int               devid,
      int               strmid);
```

Draws a bitmap on the video preview Window. VMR-9 does not support this in 64-bit mode. If your application requires this feature and the application must be 64-bits, use VMR-7 instead. The use of this function in C++ is shown in the demo application in the SDK.

`hdcBMP`
handle to bitmap to draw.

`src`
source rectangle

`dst`
destination rectangle in normalized coordinates

`alpha`
alpha value (0.0f to 1.0f)

`devids`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

```
MID2253_API int __stdcall S2253_DrawBitmapColorRef (
```

```
      HDC               hdcBMP,
      RECT              *src,
      NORMALIZEDRECT    *dst,
      float             alpha,
      COLORREF          clrSrcKey
      int               devid,
      int               strmid);
```

Draws a bitmap on the video preview Window. Colorref always for transparency. Note that VMR-9 does not support this in 64-bit mode. If your application requires this feature and the application must be 64-bits, use VMR-7 instead. The use of this function in C++ is shown in the demo application in the SDK.

`hdcBMP`
handle to bitmap to draw.

`src`
source rectangle

`dst`
destination rectangle in normalized coordinates

`alpha`
alpha value (0.0f to 1.0f)

`clrSrcKey`
color transparency key.

`devids`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

Mode Control Functions

S2253_SetVidSys

```
MID2253_API int __stdcall S2253_SetVidSys (
      MID2253_VIDSYS   vidsys,
      int              devid);
```

Sets the input video system (NTSC, PAL). Note: applies to both streams.

`vidsys`
video system enumerated type (see `mid2253types.h`).

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetVidSys

```
MID2253_API int __stdcall S2253_GetVidSys (
      MID2253_VIDSYS   *vidsys,
      int              devid);
```

Gets the input video system (NTSC, PAL).

`vidsys`
pointer to video system enumerated type (see `mid2253types.h`).

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetLevel

```
MID2253_API int __stdcall S2253_SetLevel (
      int              param,
      unsigned char    value,
      int              devid);
```

Sets brightness, contrast, saturation and hue of the captured video. Note: applies to both streams.

`param`
defines the parameter to set (`MID2253_LEVEL_CONTRAST, MID2253_LEVEL_BRIGHTNESS, MID2253_LEVEL_SATURATION, MID2253_LEVEL_HUE, MID2253_LEVEL_HCENTER, MID2253_LEVEL_VCENTER, MID2253_LEVEL_FIELDMODE, MID2253_LEVEL_AUTOGAIN, MID2253_LEVEL_AUTOBRIGHTNESS, MID2253_LEVEL_COMBFILTER, MID2253_LEVEL_LUMAFILTER, MID2253_LEVEL_CHROMAFILTER`). See `mid2253types.h` for definitions.

`value`
defines the value of selected parameter.

`devid`

device id in the system (use 0 with a single board installed).

S2253_GetLevel

```
MID2253_API int __stdcall S2253_GetLevel (
      int               param,
      unsigned char     *value,
      int               devid);
```

Retrieves current brightness, contrast, saturation and hue settings.

`param`
defines the parameter to get (`MID2253_LEVEL_CONTRAST, MID2253_LEVEL_BRIGHTNESS, MID2253_LEVEL_SATURATION, MID2253_LEVEL_HUE, MID2253_LEVEL_HCENTER, MID2253_LEVEL_VCENTER, MID2253_LEVEL_FIELDMODE, MID2253_LEVEL_AUTOGAIN, MID2253_LEVEL_AUTOBRIGHTNESS)`. See `mid2253types.h` for definitions.

`value`
pointer to returned value of selected parameter.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetAutoBrightness

```
MID2253_API int __stdcall S2253_SetAutoBrightness (
      int               bAuto,
      int               devid);
```

Deprecated, but kept for backward compatibility. Please use S2253_SetLevel with S2253_LEVEL_AUTOBRIGHTNESS parameter. Sets or disables automatic brightness on the device.

`value`
on(1) or off(0).

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetAutoGain

```
MID2253_API int __stdcall S2253_SetAutoBrightness (
      int               bAuto,
      int               devid);
```

Deprecated, but kept for backward compatibility. Please use S2253_SetLevel with S2253_LEVEL_AUTOGAIN parameter. Sets or disables automatic gain control on the device.

`value`
on(1) or off(0).

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetAutoBrightness

```
MID2253_API int __stdcall S2253_GetAutoBrightness (
      int               *bAuto,
      int               devid);
```

Deprecated, but kept for backward compatibility. Please use S2253_GetLevel with S2253_LEVEL_AUTOBRIGHTNESS parameter. Retrieves automatic brightness setting.

`value`
on(1) or off(0).

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetAutoGain

```
MID2253_API int __stdcall S2253_GetAutoGain (
      int               *bAuto,
      int               devid);
```

Deprecated, but kept for backward compatibility. Please use S2253_GetLevel with S2253_LEVEL_AUTOGAIN parameter. Retrieves automatic gain setting.

`value`
on(1) or off(0).

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetImageSize

```
MID2253_API int __stdcall S2253_SetImageSize (
      int               width,
      int               height,
      int               devid,
      int               strmidx);
```

Sets the image size (resolution). The following restrictions apply to the values.

Horizontal (`width`) – minimum 128, maximum 768. Must be a multiple of 16. A value of 768 may be used to obtain proper aspect ratio when capturing PAL. However, this size is achieved by upsampling an image captured at 720 pixel resolution. Using a

value of 768 is likely to decrease the frame capture rate of the uncompressed stream by up to 10%. Recommended values: 320, 640, 704, 720.

Vertical (`height`) – minimum 96, maximum 576. Must be a multiple of 16. Recommended values: 240, 480 for NTSC, 288, 576 for PAL.

If an invalid `width` or `height` value is passed to the function, the firmware will correct it to the nearest legitimate value. It is strongly recommended to follow a call to `S2253_SetImageSize` with a call to `S2253_GetImageSize` (see below), which will return the actual values set in the firmware, and use those values in the application.

Please note that S2253_SetImageSize must be called before the stream is started. The image size may not be changed until the stream is stopped.

`width`
width of the image, pixels.

`height`
height of the image, pixels.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetImageSize

```
MID2253_API int __stdcall S2253_GetImageSize (
      int              *width,
      int              *height,
      int              devid,
      int              strmidx);
```

Retrieves currently set image size (resolution). The values may not match those passed by `S2253_SetImageSize` function, in case those were invalid. If the size changes, it will change after the stream is started. Therefor it is best to call S2253_GetImageSize after the stream is running.

`width`
pointer to the variable receiving the width of the image.

`height`
pointer to the variable receiving the height of the image.

`devid`

device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetStreamType

```
MID2253_API int __stdcall S2253_SetStreamType(
      MID2253_STREAMTYPE    stype,
      int                   devid,
      int                   strmidx);
```

Sets the required stream type (compression).

`stype`
One of supported stream types.

MID2253_STYPE_MPEG4
MID2253_STYPE_H264
MID2253_STYPE_MJPEG
MID2253_STYPE_UYVY
MID2253_STYPE_Y800 (monochrome Y8)
MID2253_STYPE_M4TS (MPEG4 in MPEG TS stream)
MID2253_STYPE_H4TS (H.264 in MPEG TS stream)
MID2253_STYPE_YUY2
MID2253_STYPE_RGB565 (*only available for callback mode)
MID2253_SYTPE_RGB24

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetStreamType

```
MID2253_API int __stdcall S2253_GetStreamType(
      MID2253_STREAMTYPE    *stype,
      int                   devid,
      int                   strmidx);
```

Gets the currently set stream type (compression).

`stype`

One of supported stream types.

MID2253_STYPE_MPEG4
MID2253_STYPE_H264
MID2253_STYPE_MJPEG
MID2253_STYPE_UYVY
MID2253_STYPE_Y800 (monochrome Y8)
MID2253_STYPE_M4TS (MPEG4 in MPEG TS stream)
MID2253_STYPE_H4TS (H.264 in MPEG TS stream)
MID2253_STYPE_YUY2
MID2253_STYPE_RGB565 (*only available for callback mode)
MID2253_SYTPE_RGB24

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetH264Profile

```
MID2253_API int __stdcall S2253_SetH264Profile(
      MID2253_H264_PROFILE  profile,
      int                   devid,
      int                   strmidx);
```

Changes the H.264 profile for H.264 encoding modes. Default is high profile.

`profile`
MID2253_H264_BASELINE (= 66)
MID2253_H264_MAIN (= 77)
MID2253_H264_HIGH(= 100, default setting)

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetH264Profile

```
MID2253_API int __stdcall S2253_GetH264Profile(
      MID2253_H264_PROFILE  *profile,
      int                   devid,
      int                   strmidx);
```

Retrieves the current H.264 profile for H.264 encoding modes.

`profile`

MID2253_H264_BASELINE (= 66)
MID2253_H264_MAIN (= 77)
MID2253_H264_HIGH(= 100, default setting)

`devid`

device id in the system (use 0 with a single board installed).

`strmidx`

stream index (0 or 1).

S2253_SetH264Level

```
MID2253_API int __stdcall S2253_SetH264Level(
      MID2253_H264_LEVEL    level,
      int                   devid,
      int                   strmidx);
```

Changes the H.264 level for H.264 encoding modes. Default is 4.0.

`level`

MID2253_H264_1_0
MID2253_H264_1B
MID2253_H264_1_1
MID2253_H264_1_2
MID2253_H264_1_3
MID2253_H264_2_0
MID2253_H264_2_1
MID2253_H264_2_2
MID2253_H264_3_0
MID2253_H264_3_1
MID2253_H264_3_2
MID2253_H264_4_0
MID2253_H264_4_1
MID2253_H264_4_2
MID2253_H264_5_0

`devid`

device id in the system (use 0 with a single board installed).

`strmidx`

stream index (0 or 1).

S2253_GetH264Level

```
MID2253_API int __stdcall S2253_GetH264Level(
      MID2253_H264_LEVEL    *level,
```

```
    int                 devid,
    int                 strmidx);
```

Retrieves the current H.264 level for H.264 encoding modes.

`level`
- MID2253_H264_1_0
- MID2253_H264_1B
- MID2253_H264_1_1
- MID2253_H264_1_2
- MID2253_H264_1_3
- MID2253_H264_2_0
- MID2253_H264_2_1
- MID2253_H264_2_2
- MID2253_H264_3_0
- MID2253_H264_3_1
- MID2253_H264_3_2
- MID2253_H264_4_0
- MID2253_H264_4_1
- MID2253_H264_4_2
- MID2253_H264_5_0

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetMpegAspectRatio

```
MID2253_API int __stdcall S2253_SetMpegAspectRatio(
    int             val,
    int             devid,
    int             strmidx);
```

Sets the aspect ratio for the recorded MPEG stream.

`val`
0 – 1:1 (square), 1 – 4:3, 2 – 16:9

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetVcrMode

```
MID2253_API int __stdcall S2253_SetVcrMode(
      int             bVCR,
      int             devid);
```

Sets the VCR mode in DirectShow. If turned off, mode is auto-detected. Applies to video input only. Normally users of the SDK do not need to set this setting.

`bVCR`
0 – no VCR mode (default), 1 – VCR mode on.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetAudioDelay

```
MID2253_API int __stdcall S2253_SetAudioDelay(
      int             val,
      int             devid,
      int             strmidx);
```

Sets the audio delay between video stream and audio stream. Normally users of the SDK do not need to set this setting. Changing this setting may affect the A/V synchronization.

`val`
audio delay for stream, range: -500ms to 500ms. Default 0ms.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetBitrate

```
MID2253_API int __stdcall S2253_SetBitrate(
      int             bitrate,
      int             devid,
      int             strmidx);
```

Sets the bitrate for H.264 or MPEG4 encoding, in kilobits per second (kbps). Allowed ranges: 100-20000 kbps for MPEG-4 and H.264.

It is recommended to stay above 700 kbps for full size (640x480 and larger) resolutions.

The target bitrate rate control will not be used when FixedQP is enabled.

`bitrate`
bitrate in kbps.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetBitrate

```
MID2253_API int __stdcall S2253_GetBitrate(
      int              *bitrate,
      int              devid,
      int              strmidx);
```

Gets the current bitrate settings, in kilobits per second (kbps).

`bitrate`
a pointer to the variable receiving the bitrate value in kbps.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetFixedQP

```
MID2253_API int __stdcall S2253_SetFixedQP (
      int              intraframeqp,
      int              interpframeqp,
      int              devid,
      int              strmidx);
```

Sets the constant quality (fixed QP) bitrate mode for the H.264 compression. The QP parameters are set to -1 by default, indicating that the bitrate parameter should be used to rate control the stream. Otherwise the QP parameters will control the quality of the stream, and the bitrate parameter is not used for rate control. Lower numbers produce better quality (higher bitrate) and higher numbers produce worse quality (lower bitrate). The FixedQP mode can be used to produce variable bitrate based on image complexity, but upper bound of video bitrate is not restricted for very complex images.

`intraframeqp`
Intra frame quality factor, range 0 to 42. FixedQP is disabled when -1.

`interpframeqp`
Inter P-frame quality factor, range 0 to 45. FixedQP is disabled when -1.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetFixedQP

```
MID2253_API int __stdcall S2253_GetFixedQP (
      int               *intraframeqp,
      int               *interpframeqp,
      int               devid,
      int               strmidx);
```

Retrieves the constant quality (Fixed QP) factors for H.264 encoding.

`intraframeqp`
a pointer to a variable receiving Intra frame quality factor value.

`interpframeqp`
a pointer to a variable receiving Inter P-frame quality factor value.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetJpegQ

```
MID2253_API int __stdcall S2253_SetJpegQ (
      int               q,
      int               devid,
      int               strmidx);
```

Sets the JPEG quality factor for motion JPEG encoding. The allowed values are 2-97. Higher values result in higher image quality and larger image sizes.

`q`
jpeg quality factor.

`devid`

device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetJpegQ

```
MID2253_API int __stdcall S2253_GetJpegQ (
      int               *q,
      int               devid,
      int               strmidx);
```

Retrieves the JPEG quality factor for motion JPEG encoding.

`q`
a pointer to a variable receiving JPEG quality factor value.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetIDR

```
MID2253_API int __stdcall S2253_SetIDR (
      int               val,
      int               devid,
      int               strmidx);
```

Sets the IDR (instantaneous decoding refresh) frame frequency (with respect to GOPs) for H.264 streams. Does not apply to stream types other than H.264. If val is set to 0 (default setting), there is only one IDR at the start of the stream. If set to 1, there is an IDR every GOP. If val is set to 2, there is an IDR every other GOP.

`val`
frequency of IDRs in H.264 stream.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetGopSize

```
MID2253_API int __stdcall S2253_SetGopSize (
      int               gopsize,
```

```
      int               devid,
      int               strmidx);
```

Sets the GOP (group of pictures) size for MPEG compressed streams (MPEG4, H.264). Use 0 for default.

`gopsize`
gop size. (Use 0 for default).

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetGopSize

```
MID2253_API int __stdcall S2253_GetGopSize (
      int               *gopsize,
      int               devid,
      int               strmidx);
```

Returns the current GOP (group of pictures) size for MPEG compressed streams (MPEG4, H.264).

`gopsize`
gop size.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetClock

```
MID2253_API int __stdcall S2253_SetClock (
      MID2253_USER_CLOCK      *pclk,
      int                     devid);
```

Sets the clock on the device for OSD timestamps. Use Unix style GMT time. See the demo application for an example. Set before starting any streams on the device. Must not be set during streaming.

`pclk`
a pointer to the current clock value.

`devid`

device id in the system (use 0 with a single board installed).

S2253_SetFrameCount

```
MID2253_API int __stdcall S2253_SetFrameCount (
      int               count,
      int               devid);
```

Sets the frame counter value.

`count`
frame counter value

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetFrameCount

```
MID2253_API int __stdcall S2253_SetFrameCount (
      int               *count,
      int               devid);
```

Gets the frame counter value.

`count`
frame counter value.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetInterpolateMode

```
MID2253_API int __stdcall S2253_SetInterpolateMode (
      int               val,
      int               devid);
```

Sets interpolation mode used to reduce motion artifacts due to interlacing. When interpolation is turned on, one field of every captured frame is dropped and recreated by interpolating neighboring lines of the other field. The setting affects both video streams.

`val`
0 – interpolation is off; 1 – interpolation is on.

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetInterpolateMode

```
MID2253_API int __stdcall S2253_GetInterpolateMode (
      int               *val,
      int               devid);
```

Retrieves interpolate mode.

`val`
a pointer to the variable receiving interpolation setting.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetDeinterlaceMode

```
MID2253_API int __stdcall S2253_SetDeinterlaceMode (
      int               val,
      int               devid);
```

Sets deinterlace mode used to reduce motion artifacts due to interlacing, but uses an algorithm that is visually more appealing than interpolation. The setting affects both video streams. The deinterlace mode takes precedence over interpolate mode. The image resizer cannot be used while deinterlacing is enabled, so the image will be cropped instead. Image flipping and rotation are also not avaiable. The deinterlacing adds one frame extra latency to the video processing pipeline. The pixel aspect ratio will need to match the source video, so be sure to set S2253_SetMpegAspectRatio to non-square pixels.

`val`
0 – deinterlacing is off; 1 – deinterlacing is on.

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetDeinterlaceMode

```
MID2253_API int __stdcall S2253_GetDeinterlaceMode (
      int               *val,
      int               devid);
```

Retrieves deinterlace mode.

`val`
a pointer to the variable receiving deinterlace setting.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetInputCrop

```
MID2253_API int __stdcall S2253_SetInputCrop (
      int               left,
      int               top,
      int               width,
      int               height,
      int               devid);
```

Sets the cropping window of the captured input. The setting affects both video streams. The height will be limited automatically to the video standard, so for full-frame don't need to specify 480 for NTSC and 576 for PAL, just use 576 for both. A smaller window should take the video standard into account, for example 180,120,360,240 for NTSC and 180,144,360,288 for PAL.

`left, top, width, height`
specifies the position of the cropping window. Default=8,0,704,576

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetInputCrop

```
MID2253_API int __stdcall S2253_GetInputCrop (
      int               *left,
      int               *top,
      int               *width,
      int               *height,
      int               devid);
```

Retrieves input crop window.

`left, top, width, height`
pointers to the variables receiving crop parameters.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetRecordMode

```
MID2253_API int __stdcall S2253_SetRecordMode (
      MID2253_RECMODE   recmode,
      int               devid,
      int               strmidx);
```

Controls whether audio is recorded or not. Also controls the multiplex (mux) format. MID2253_RECMODE_VIDEO is multiplexed video only. If the stream type is MPEG-4 or H.264, the format will be MP4. Otherwise the format is AVI.

MID2253_RECMODE_AV is multiplexed audio and video. The mux format is the same as for MID2253_RECMODE_VIDEO.

MID2253_RECMODE_VES is video only elementary stream. There is no mux or container format. This mode applies to MPEG-4 or H.264 only. If the stream type is a different stream type, it will be formatted as AVI.

`recmode`
MID2253_RECMODE_VIDEO, MID2253_RECMODE_AV or MID2253_RECMODE_VES.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetRecordMode

```
MID2253_API int __stdcall S2253_GetRecordMode (
      MID2253_RECMODE   *recmode,
      int               devid,
      int               strmidx);
```

Returns current recording mode setting.

`recmode`
pointer to returned value MID2253_RECMODE_VIDEO, MID2253_RECMODE_AV, or MID2253_RECMODE_VES.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetMp4Mode

```
MID2253_API int __stdcall S2253_SetMp4Mode (
      MID2253_MP4MODE   mp4mode,
      int               devid,
      int               strmidx);
```

Sets the .mp4 mux format for stream types H264 or MPEG4 (with record mode MID2253_RECMODE_VIDEO or MID2253_RECMODE_AV). Files recorded using

MID2253_MP4MODE_STANDARD will be playable by most standard media players. MID2253_MP4MODE_STREAMABLE uses a fragmented MP4 format that allows streaming (instead of an index table at the end of the file). The streamable format, however, is supported by fewer media players at this time.

`mp4mode`
MID2253_MP4MODE_STANDARD or MID2253_MP4MODE_STREAMABLE.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetMp4Mode

```
MID2253_API int __stdcall S2253_GetRecordMode (
      MID2253_MP4MODE   *mp4mode,
      int               devid,
      int               strmidx);
```

Returns current .mp4 mux/container format.

`mp4mode`
pointer to returned value MID2253_MP4MODE_STANDARD or MID2253_MP4MODE_STREAMABLE.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetAudioEncoding

```
MID2253_API int __stdcall S2253_SetAudioEncoding (
      MID2253_AUDENC    aenc,
      int               devid,
      int               strmidx);
```

Sets the audio encoding type. This setting only applies to files recorded with audio with stream types of UYVY, MJPEG or Y800. The audio encoding for files recorded as MPEG4 or H264 will always be AAC. This function has no effect if the stream is set to MPEG4 or H264.

`aenc`

MID2253_AUDENC_PCM, MID2253_AUDENC_G711_ULAW, MID2253_AUDENC_G711_ALAW.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetAudioEncoding

```
MID2253_API int __stdcall S2253_GetAudioEncoding (
      MID2253_AUDENC    *aenc,
      int               devid,
      int               strmidx);
```

Returns current audio encoding setting.

`aenc`
MID2253_AUDENC_PCM, MID2253_AUDENC_G711_ULAW, MID2253_AUDENC_G711_ALAW.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetAudioBitrate

```
MID2253_API int __stdcall S2253_SetAudioBitrate (
      MID2253_AUDIO_BITRATE   audbr,
      int                     devid,
      int                     strmidx);
```

Sets the audio encoding bitrate. This setting only applies to files recorded with AAC audio. The audio encoding must be set before the stream is started. On the fly changes are not supported.

`audbr`
MID2253_AUDBR_96 (96kbps), MID2253_AUDBR_128 (128kbps), MID2253_AUDBR_192 (192kbps), MID2253_AUDBR_224 (224kbps), MID2253_AUDBR_256 (256kbps).

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetAudioBitrate

```
MID2253_API int __stdcall S2253_GetAudioBitrate (
      MID2253_AUDIO_BITRATE   *audbr,
      int                     devid,
      int                     strmidx);
```

Returns current audio bitrate setting.

Sets the audio encoding bitrate. This setting only applies to files recorded with AAC audio. The audio encoding must be set before the stream is started. On the fly changes are not supported.

`audbr`
MID2253_AUDBR_96 (96kbps), MID2253_AUDBR_128 (128kbps), MID2253_AUDBR_192 (192kbps), MID2253_AUDBR_224 (224kbps), MID2253_AUDBR_256 (256kbps).

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetAudioInput

```
MID2253_API int __stdcall S2253_SetAudioInput (
      MID2253_AUDIO_INPUT     input,
      int                     devid)
```

Sets the audio input. This setting applies to both streams and the separate audio stream (S2253_StartAudioPreview).

`input`
MID2253_AUDIO_MIC (microphone), MID2253_AUDIO_LINE (line).

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetAudioInput

```
MID2253_API int __stdcall S2253_GetAudioInput (
      MID2253_AUDIO_INPUT     input,
```

```
      int                       devid)
```

Gets the current audio input setting.

`input`
MID2253_AUDIO_MIC (microphone), MID2253_AUDIO_LINE (line).

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetAudioGain

```
MID2253_API int __stdcall S2253_SetAudioGain (
      BOOL        bAGC,
      int         gain,
      int         devid)
```

Sets the audio input gain. If bAGC is TRUE, the gain setting is ignored and automatic gain control is used. If bAGC is FALSE, the gain is manual. This setting applies to both streams and the separate audio stream (S2253_StartAudioPreview).

`bAGC`
Turns AGC on or off (FALSE).

`gain`
Manual gain when AGC off. Value from 0 to 119 in steps of 0.5dB. 0=0dB, 1=0.5dB, 119= 59.5dB.

`devid`
device id in the system (use 0 with a single board installed).

S2253_GetAudioGain

```
MID2253_API int __stdcall S2253_GetAudioGain (
      BOOL        bAGC,
      int         gain,
      int         devid)
```

Retrieves the audio input gain parameters.

`bAGC`
pointer to AGC control setting

`gain`
pointer to manual audio gain.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetAudioChannels

```
MID2253_API int __stdcall S2253_SetAudioChannels (
      MID2253_AUDIO_CHANNELS  chanmode,
      int                     devid,
      int                     strmidx)
```

Sets the current audio channel setting for the specified stream.

`chanmode`
MID2253_AUDCH_MONO_MIXED (left and right channels mixed)
MID2253_AUDCH_STEREO (default)
MID2253_AUDCH_MONO_LEFT (left channel only)
MID2253_AUDCH_MONO_RIGHT (right channel only)

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetAudioChannels

```
MID2253_API int __stdcall S2253_GetAudioChannels (
      MID2253_AUDIO_CHANNELS  chanmode,
      int                     devid,
      int                     strmidx)
```

Gets the current audio channel setting for the specified stream.

`chanmode`
MID2253_AUDCH_MONO_MIXED (left and right channels mixed)
MID2253_AUDCH_STEREO (default)
MID2253_AUDCH_MONO_LEFT (left channel only)
MID2253_AUDCH_MONO_RIGHT (right channel only)

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_FlipImage

```
MID2253_API int __stdcall S2253_FlipImage (
      BOOL              bFlipV,
```

```
      BOOL              bFlipH,
      int               devid,
      int               strmidx);
```

Set the image flipping feature, in the vertical and/or horizontal direction. When both bFlipV and bFlipH are set to TRUE, the image is effectively rotated 180 degrees.

`bFlipV`
Flip image vertically.

`bFlipH`
Flip image horizontally.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_LowLatencyPreview

```
MID2253_API int __stdcall S2253_LowLatencyPreview (
      BOOL              bON,
      int               devid,
      int               strmidx);
```

This setting controls the preview display latency on a PC. Normally, DirectShow(the API used by Windows for Video display) will buffer the displayed video by a small amount to make it smooth. DirectShow, however, has a setting to display the images as soon as they are available. This reduces the latency at the expense of possibly choppy or unsmooth video. Please note that this setting has NO effect on the latency of the stream captured by the hardware. For lower capture latency, see the section Ultra-low Latency Preview earlier in this manual. This function should be called before S2253_StartPreview if it is used.

`bON`
on or off setting.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_SetUserData

```
MID2253_API int __stdcall S2253_SetUserData (
```

```
    char            *data,
    int             len,
    int             interval,
    int             devid,
    int             strmidx);
```

Inserts user data into the stream. Applies to H.264, MPEG-4 only. User data can be a string(company name for instance) or binary data(GPS data for instance).

`data`
data to insert

`len`
length of data

`interval`
frequency to add data to the stream in terms of GOPs. Use 0 to insert data once (useful for GPS data). Use any other value, N, to insert data after every Nth GOP (useful for water-marking the stream with a company name or Copyright). Eg. If interval = 3, the user data will be inserted every 3rd GOP.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_EnableClosedCaptions

```
MID2253_API int __stdcall S2253_EnableClosedCaptions (
    int             enabled,
    int             devid,
    int             strmidx);
```

Enables capturing and insertion of closed-caption data into the stream. Applies to H.264 only. When enabled, user data insertion will be overridden.

`enabled`
on or off setting.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_GetDecodeStatus

```
MID2253_API int __stdcall S2253_EnableClosedCaptions (
```

```
      int               *status,
      int               devid);
```

Returns current decode statu.

`status`
0 –idle, 1 – decoding, otherwise error code.

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetOsd

```
MID2253_API int __stdcall S2253_SetOSD (
      MID2253_OSD_TYPE  osdtype,
      MID2253_OSD_DATA  *osddata,
      int               devid,
      int               strmidx);
```

Controls on-screen display (OSD).

Currently OSD is limited to one line of text (80 or 160 characters maximum) for MID2253_OSDTYPE_TEXT and MID2253_OSDTYPE_LONGTEXT. More text regions are supported with MID2253_OSDTYPE_STYLEDTEXT.

For MID2253_OSDTYPE_TEXT and MID2253_OSDTYPE_LONGTEXT, there is the possibility to insert newlines, date/time and a frame counter. Use the following codes to display extra information in the text:

^d (date)
^t (time)
^n (newline)
^c (frame counter)
^p (GPS latitude, 2253P only)
^l (GPS longitude, 2253P only)
^a (GPS altitude, 2253P only)
^m (GPS UTC date, 2253P only)
^u (GPS UTC time, 2253P only)
^s (GPS speed in knots, 2253P only)
^h (GPS course heading, 2253P only)
^e (encoder 0 count, 2253P only)
^f (encoder 1 count, 2253P only)

Note: not applicable to the MID2253_OSDTYPE_STYLEDTEXT type.

Styled text (MID2253_OSDTYPE_STYLEDTEXT) allows the use of true type fonts, Unicode, and more regions of text. Please note that excessive text overlay may impact the frame rate on Streams A and B. An example of using styled text is shown in the Appendix at the end of this document. See the S2253_StreamOverlay function for notes on overlay performance which affect this style.

Rectangles may be drawn using MID2253_OSDTYPE_RECT.

A WIN32 BMP may be drawn using MID2253_OSDTYPE_BMP. It is an efficient way to overlay simple graphics on Streams A and B. Note: the pixels will be converted to grayscale and black shaded pixels only. To send full color BMP or PNG images, use the SetOverlay or StreamOverlay functions instead.

To control the stream on which the overlay is displayed, the various MID2253_OSD_ structs have a osdChan parameter, which takes precedence over the strmidx argument, and takes the following values:

0 – stream A
1 – stream B
2 – output stream
3 – both A and B, and output, simultaneously

When displaying complex overlays on multiple streams simultaneously, it is recommended to use option 3, which allows the board to draw the overlays once instead of individually for each stream. This results in efficient overlay processing and can avoid running into performance limitations.

`osdtype`
OSD type. Use `MID2253_OSDTYPE_TEXT` (80 char limit), `MID2253_OSDTYPE_LONGTEXT` (160 char limit), `MID2253_OSDTYPE_STYLEDTEXT`, `MID2253_OSDTYPE_RECT`, or `MID2253_OSDTYPE_BMP`.

`osddata`
A pointer to the structure of `MID2253_OSD_DATA` type. Please see `mid2253types.h` for the description.

`devid`
device id in the system (use 0 with a single board installed).

`strmidx`
stream index (0 or 1).

S2253_StreamOverlay

```
MID2253_API int __stdcall S2253_StreamOverlay (
      MID2253_OVL_STRUCT      *ovl,
```

```
    unsigned char           *data,
    int                     devid,
    int                     strmidx);
```

Same as S2253_SetOverlay (below), but for input streams, and the transparent field behaves differently than the S2253_SetOverlay function. When transparent is used for values 0 to 7, the overlay is rendered in grayscale. Important note: too many or too large overlays with transparent parts may overload the processor on the board. For this reason it is best to limit the number or size of overlays with transparency.

When transparent=8, the overlay is drawn in color as a solid rectangle, without any transparency effect, and is hardware-accelerated. Use this mode if performance is a concern.

`ovl`

pointer to overlay structure to update. See mid2253types.h.

MID2253_OVL_STRUCT

id: unique id representing this overlay.

0 to 15 for Streams A&B, 0 to 255 for Output

transparent: transparent field. (0 transparent to 7 opaque, 8 solid)

update: update field. Mask of the following:

MID2253_OVERLAY_UPDATE_DISPLAY,

MID2253_OVERLAY_UPDATE_TRANSPARENT,

MID2253_OVERLAY_UPDATE_POSITION

reserved: future use

xOffset: x offset from left side

yOffset: y offset from the top

length: length of data

`strmidx`

stream index: 0(A), 1(B), 2(output), 3(combined A/B/output)

`devid`

device id in the system (use 0 with a single board installed).

S2253_SetOverlay

```
MID2253_API int __stdcall S2253_SetOverlay (
      MID2253_OVL_STRUCT      *ovl,
      unsigned char           *data,
      int                     devid);
```

This function controls overlays on the physical video output channel. It is not used for overlays on the encoded or captured video stream. It uploads an image to be overlaid on the video output frames. The device remembers the id where each image was placed, allowing it to be moved later.

To move an existing overlay image, set the id, length to zero, the update field to MID2253_OVERLAY_UPDATE_POSITION and the x and y offets to the new position. Due to hardware limitations, the x offset may be limited to a multiple of 2.

The transparent setting allows the overlay to have a transparency effect, allowing the underlying video to be mixed with the image. When transparent is set to “8”, the alpha channel in a 32-bit PNG or BMP will be used.

To hide an existing image, update the overlay (with specified id) with the transparent field set to zero.

To display images, use the MID2253_OVERLAY_UPDATE_DISPLAY mask in the ovl->update field. When updating multiple overlays at once, it is only necessary (and more efficient to) set MID2253_OVERLAY_UPDATE_DISPLAY for the last “id” to update.

Image overlays are destructive; moving or overlapping images will cause the previous image to be overwritten by the rectangular region of the updated image, regardless of the transparency setting.

Please see the demo app for an example on how to use the output overlay.

`ovl`

pointer to overlay structure to update. See mid2253types.h.

MID2253_OVL_STRUCT

id: unique id representing this overlay. (0 to 255)

transparent: transparent field. (0 transparent to 7 opaque, 8 alpha)

update: update field. Mask of the following:

MID2253_OVERLAY_UPDATE_DISPLAY,

MID2253_OVERLAY_UPDATE_TRANSPARENT,

MID2253_OVERLAY_UPDATE_POSITION

reserved: future use

xOffset: x offset from left side

yOffset: y offset from the top

length: length of data

`strmidx`
stream index (0 or 1).

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetOutputMode

```
MID2253_API int __stdcall S2253_SetOutputMode (
      MID2253_OUTPUT_MODE     mode,
      int                     devid);
```

This function controls what is displayed on the physical video output channel. Please note that when decoding (S2253_StartDecode), the output is automatically set to MID2253_OUTPUT_STREAM.

`mode`
MID2253_OUTPUT_IDLE (no video output), MID2253_OUTPUT_PASSTHRU (capture passthru), MID2253_OUTPUT_COLORBARS, MID2253_OUTPUT_FLASH (displays 1 white frame then switches to idle), MID2253_OUTPUT_STREAM (normally unused, for decode).

`devid`
device id in the system (use 0 with a single board installed).

S2253_SetOutputVGA

```
MID2253_API int __stdcall S2253_SetOutputVGA (
      BOOL                    enable,
      int                     devid);
```

Enables or disables VGA mode (640 pixels wide) on the composite output.

`enable`
set to TRUE to turn on VGA mode, FALSE to turn off.

`devid`
device id in the system (use 0 with a single board installed).

S2253P_ReadOnline

```
MID2253_API int __stdcall S2253P_ReadOnline (
      int                     devid,
      int                     *online);
```

This function reads the online status of the 2253P-specific hardware, and should be used to verify the system is online before using any other 2253P-specific functions.

`devid`
device id in the system (use 0 with a single board installed).

`online`
pointer to variable receiving online status: 1=online, 0=other.

S2253P_ReadVersion

```
MID2253_API int __stdcall S2253P_ReadVersion (
      int                     devid,
      int                     *version);
```

This function reads the version of the firmware on the 2253P-specific hardware.

`devid`
device id in the system (use 0 with a single board installed).

`version`
pointer to variable receiving version information.

S2253P_ReadComstat

```
MID2253_API int __stdcall S2253P_ReadComstat (
      int                     devid,
      void                    *comstat);
```

This function reads the 2253P communications status registers.

`devid`
device id in the system (use 0 with a single board installed).

`comstat`

pointer to an 6-element array of 16-bit unsigned integer counts for the following:

[0] framing errors
[1] overrun errors
[2] buffer overflow
[3] gps framing errors
[4] gps overrun errors
[5] gps buffer overflow

S2253P_SetSuspend

```
MID2253_API int __stdcall S2253P_SetSuspend (
      int                     devid,
      int                     suspend);
```

This function controls whether the 2253P-specific hardware should be held in reset during when the USB device is suspended (during host pc suspend). If the hardware is suspended, then encoder changes during suspend will be lost.

`devid`
device id in the system (use 0 with a single board installed).

`suspend`
1=suspend holds device in reset
0=device continues to run during suspend

S2253P_EncoderReset

```
MID2253_API int __stdcall S2253P_EncoderReset (
      int                     devid,
      int                     encoderId);
```

This function resets the encoder count for the specified encoder to zero.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

S2253P_EncoderLoad

```
MID2253_API int __stdcall S2253P_EncoderLoad (
      int                     devid,
```

```
      int                     encoderId,
      int                     value);
```

This function loads a value into the encoder count for the specified encoder.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

`value`
requested value.

S2253P_EnableEncoderAsync

```
MID2253_API int __stdcall S2253P_EnableEncoderAsync (
      int                     devid,
      int                     encoderId,
      int                     enable);
```

This function controls enabling asynchronous encoder count updates internally on the board. When asynchronous mode is enabled, encoder count on OSD will update on-the-fly, and encoder reads do not require update first.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

`enable`
0=disabled, 1=enable asynchronous updates

S2253P_EncoderRead

```
MID2253_API int __stdcall S2253P_EncoderRead (
      int                     devid,
      int                     encoderId,
      S2253P_MODE             mode,
      int                     *value);
```

This function reads the encoder. This requires two transactions, an “update” request, and a “read” reply. The transactions may be performed independently, or together. Any read transaction will cause this function to block until the reply is received. When asynchronous mode is enabled for an encoder using

S2253P_EnableEncoderAsync, the “update” transaction is not required and a read should not block.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

`mode`
S2253P_MODE_UPDATE: send encoder update transaction
S2253P_MODE_READ: read encoder reply
S2253P_MODE_UPDATE_READ: both

`value`
pointer to variable receiving current value, for read transactions

S2253P_EncoderReadScaled

```
MID2253_API int __stdcall S2253P_EncoderReadScaled (
      int                   devid,
      int                   encoderId,
      S2253P_MODE           mode,
      double                *value);
```

This function reads the encoder count and multiplies the value by a scale factor. The transaction mode is the same as S2253P_EncoderRead. The scale factor is set by either S2253P_EncoderSetScale or S2253P_EncoderLoadScaled.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

`mode`
S2253P_MODE_UPDATE: send encoder update transaction
S2253P_MODE_READ: read encoder reply
S2253P_MODE_UPDATE_READ: both

`value`
pointer to variable receiving current value, for read transactions

S2253P_EncoderLoadScaled

```
MID2253_API int __stdcall S2253P_EncoderLoadScaled (
```

```
      int                     devid,
      int                     encoderId,
      double                  value);
```

This function reads the encoder count and calculates the scale factor required to multiply the current count equal to value. For example, to perform a two-point encoder calibration, set the encoder shaft to the 0 position and call S2253P_EncoderReset. Now rotate the encoder shaft to 100 units, and call S2253P_EncoderLoadScaled(devid, encid, 100). The function will calculate the scale factor internally, and calls to S2253P_EncoderReadScaled will return a scaled value in unit terms. The '^e' and '^f' OSD codes will also use the scaled value.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

`value`
the desired position in units at the current encoder count

S2253P_EncoderSetScale

```
MID2253_API int __stdcall S2253P_EncoderSetScale (
      int                     devid,
      int                     encoderId,
      double                  scale);
```

This function set the scale factor directly to be used with the S2253P_EncoderReadScaled function. The '^e' and '^f' OSD codes will also use the scaled value.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

`scale`
the scale factor for multiplying the encoder count to scaled units

S2253P_EncoderGetScale

```
MID2253_API int __stdcall S2253P_EncoderGetScale (
      int                     devid,
```

```
      int                     encoderId,
      double                  *scale);
```

This function gets the scale factor set by the S2253P_Encoder_SetScale or S2253P_EncoderLoadScaled functions.

`devid`
device id in the system (use 0 with a single board installed).

`encoderId`
encoder identifier 0 or 1.

`scale`
pointer the variable receiving the current scale factor

S2253P_GpioConfig

```
MID2253_API int __stdcall S2253P_GpioConfig (
      int                     devid,
      int                     gpioId,
      S2253P_GPIO_DIRECTION   direction);
```

This function configures a gpio with a direction.

`devid`
device id in the system (use 0 with a single board installed).

`gpioId`
gpio identifier 0 or 1.

`direction`
S2253P_GPIO_DIR_IN
S2253P_GPIO_DIR_OUT

S2253P_GpioWrite

```
MID2253_API int __stdcall S2253P_GpioWrite (
      int                     devid,
      int                     gpioId,
      int                     value);
```

This function sets a gpio output level. It must be configured as an output.

`devid`
device id in the system (use 0 with a single board installed).

`gpioId`

gpio identifier 0 or 1.

`value`
0 or 1 (active low)

S2253P_GpioRead

```
MID2253_API int __stdcall S2253P_GpioRead (
      int                     devid,
      int                     gpioId,
      int                     *value,
      S2253P_MODE             mode);
```

This function reads a gpio. This requires two transactions, an “update” request, and a “read” reply. The transactions may be performed independently, or together. Any read transaction will cause this function to block until the reply is received.

`devid`
device id in the system (use 0 with a single board installed).

`gpioId`
gpio identifier 0 or 1.

`value`
pointer to variable receiving the value: 0 or 1 (active low)

`mode`
S2253P_MODE_UPDATE: send gpio update transaction
S2253P_MODE_READ: read gpio reply
S2253P_MODE_UPDATE_READ: both

S2253P_EnableXIOAsync

```
MID2253_API int __stdcall S2253P_EnableXIOAsync (
      int                     devid,
      int                     xioId,
      int                     enable);
```

This function controls enabling asynchronous XIO updates internally on the board. When asynchronous mode is enabled, XIO reads do not require update first, and XIO triggers may be used to pause streaming.

`devid`
device id in the system (use 0 with a single board installed).

`xioId`
xio identifier 0 to 3.

`enable`
0=disabled, 1=enable asynchronous updates

S2253P_ReadXIO

```
MID2253_API int __stdcall S2253P_ReadXIO (
      int                   devid,
      int                   xioId,
      int                   *value,
      S2253P_MODE           mode);
```

This function reads the XIO. This requires two transactions, an “update” request, and a “read” reply. The transactions may be performed independently, or together. Any read transaction will cause this function to block until the reply is received. When asynchronous mode is enabled for an XIO using S2253P_EnableXIOAsync, the “update” transaction is not required and a read should not block.

`devid`
device id in the system (use 0 with a single board installed).

`xioId`
xio identifier 0 to 3.

`value`
pointer to variable receiving current value, for read transactions: 0 or 1 (active low)

`mode`
S2253P_MODE_UPDATE: send encoder update transaction
S2253P_MODE_READ: read encoder reply
S2253P_MODE_UPDATE_READ: both

S2253P_PauseConfigXIO

```
MID2253_API int __stdcall S2253P_PauseConfigXIO (
      int                   devid,
      int                   streamId,
      int                   xioId,
      S2253P_XIO_PAUSE_MODE mode);
```

This function configures a stream for pausing based on XIO events. This requires that asynchronous mode be enabled for the XIO using S2253P_EnableXIOAsync.

The stream can be paused when one of the following events occurs: rising edge, falling edge, level high, level low. If the high/low event condition is met when the stream is started, the stream will be paused and no frames delivered until the condition changes.

`devid`
device id in the system (use 0 with a single board installed).

`streamId`
Stream identifier 0=Stream A, 1=Stream B, 2=Output.

`xioId`
xio identifier 0 to 3.

`mode`
S2253P_XIO_PAUSE_MODE_DISABLE: stream will not pause
S2253P_XIO_PAUSE_MODE_RISING_EDGE
S2253P_XIO_PAUSE_MODE_FALLING_EDGE
S2253P_XIO_PAUSE_MODE_LEVEL_HIGH
S2253P_XIO_PAUSE_MODE_LEVEL_LOW

S2253P_EnableGPS

```
MID2253_API int __stdcall S2253P_EnableGPS (
      int                     devid,
      int                     enable);
```

This function enables or disables the GPS commands on the 2253P-specific hardware. When enabled, the GPS data is automatically updated on the OSD, and the following GPS functions will return data. The following codes may be used with S2253_SetOsd function to insert GPS data into the OSD text.

^p (GPS latitude)
^l (GPS longitude)
^a (GPS altitude)
^m (GPS UTC date)
^u (GPS UTC time)
^s (GPS speed in knots)
^h (GPS course heading)
^e (encoder 0 count)
^f (encoder 1 count)

`devid`
device id in the system (use 0 with a single board installed).

`enable`
0=disable 1=enable

S2253P_ReadGPSStatus

```
MID2253_API int __stdcall S2253P_ReadGPSStatus (
      int                     devid,
      int                     *status);
```

This function reads the GPS status on the 2253P-specific hardware.

`devid`
device id in the system (use 0 with a single board installed).

`status`
pointer to variable receiving the GPS status.

S2253P_ReadLatitude

```
MID2253_API int __stdcall S2253P_ReadLatitude (
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the GPS coordinate latitude into a character buffer. The buffer will be empty if no lock or location data is available yet. The returned buffer will be in the format of “ddmm.mmmmx”.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive latitude.

`size`
size of the buffer.

S2253P_ReadLongitude

```
MID2253_API int __stdcall S2253P_ReadLongitude (
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the GPS coordinate longitude into a character buffer. The buffer will be empty if no lock or location data is available yet. The returned buffer will be in the format of “ddmm.mmmmx”.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive longitude.

`size`
size of the buffer.

S2253P_ReadAltitude

```
MID2253_API int __stdcall S2253P_ReadAltitude (
      int                   devid,
      char                  *value,
      int                   size);
```

This function reads the GPS coordinate altitude into a character buffer. The buffer will be empty if no lock or location data is available yet. The returned buffer will be in the format of “nnn.nM”.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive altitude.

`size`
size of the buffer.

S2253P_ReadSpeed

```
MID2253_API int __stdcall S2253P_ReadSpeed (
      int                   devid,
      char                  *value,
      int                   size);
```

This function reads the GPS speed into a character buffer. The returned buffer will be in the format of “d.dd” in knots.

`devid`
device id in the system (use 0 with a single board installed).

`value`

pointer to character buffer to receive speed.

`size`
size of the buffer.

S2253P_ReadCourse

```
MID2253_API int __stdcall S2253P_ReadCourse (
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the GPS course heading into a character buffer. The returned buffer will be in the format of “ddddd.dd” in degrees.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive course heading.

`size`
size of the buffer.

S2253P_ReadUTCTime

```
MID2253_API int __stdcall S2253P_ReadUTCTime (
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the GPS UTC time into a character buffer. The returned buffer will be in the format of “hhmmss.sss”.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive UTC time.

`size`
size of the buffer.

S2253P_ReadUTCDate

```
MID2253_API int __stdcall S2253P_ReadUTCDate (
```

```
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the GPS UTC date into a character buffer. The returned buffer will be in the format of “ddmmyy”.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive UTC date.

`size`
size of the buffer.

S2253P_ReadSatellites

```
MID2253_API int __stdcall S2253P_ReadSatellites (
      int                     devid,
      int                     *value);
```

This function reads the number of GPS satellites in view.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to variable to receive satellite count.

S2253P_ReadLock

```
MID2253_API int __stdcall S2253P_ReadLock (
      int                     devid,
      int                     *lock);
```

This function reads the GPS lock status.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to varaible to receive GPS lock. 0=no lock, 1=locked

S2253P_ReadGPS_GGA

```
MID2253_API int __stdcall S2253P_ReadGPS_GGA (
```

```
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the most recent GPS GPGGA message into a character buffer. The returned buffer will be in the format of the NMEA 0183.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive GGA message.

`size`
size of the buffer, recommended 80.

S2253P_ReadGPS_GSA

```
MID2253_API int __stdcall S2253P_ReadGPS_GSA (
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the most recent GPS GPGSA message into a character buffer. The returned buffer will be in the format of the NMEA 0183.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive GSA message.

`size`
size of the buffer, recommended 80.

S2253P_ReadGPS_GSV

```
MID2253_API int __stdcall S2253P_ReadGPS_GSV (
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the most recent GPS GPGSV message into a character buffer. The returned buffer will be in the format of the NMEA 0183.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive GSV message.

`size`
size of the buffer, recommended 80.

S2253P_ReadGPS_RMC

```
MID2253_API int __stdcall S2253P_ReadGPS_RMC (
      int                     devid,
      char                    *value,
      int                     size);
```

This function reads the most recent GPS GPRMC message into a character buffer. The returned buffer will be in the format of the NMEA 0183.

`devid`
device id in the system (use 0 with a single board installed).

`value`
pointer to character buffer to receive RMC message.

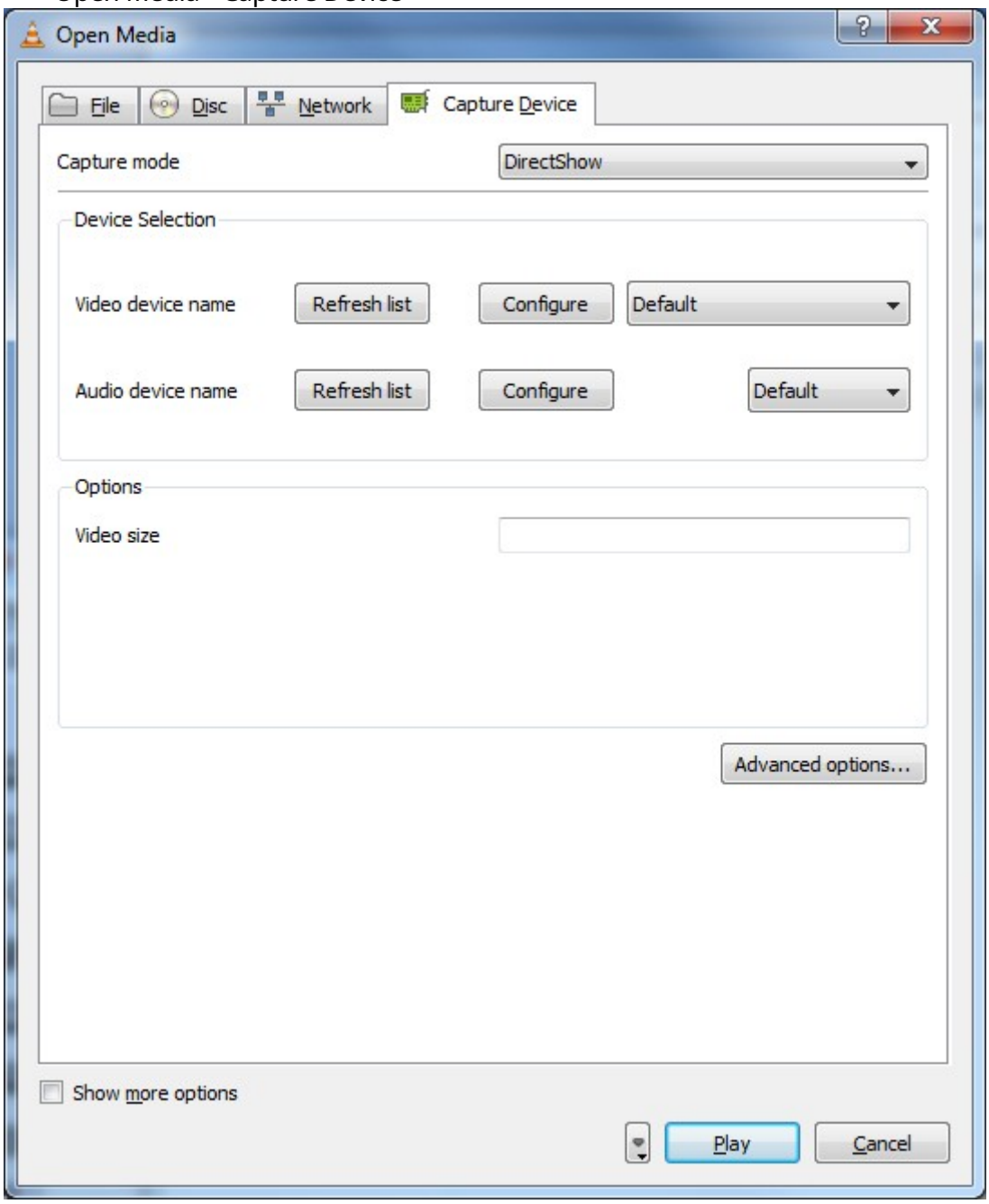
`size`
size of the buffer, recommended 80.

FAQ

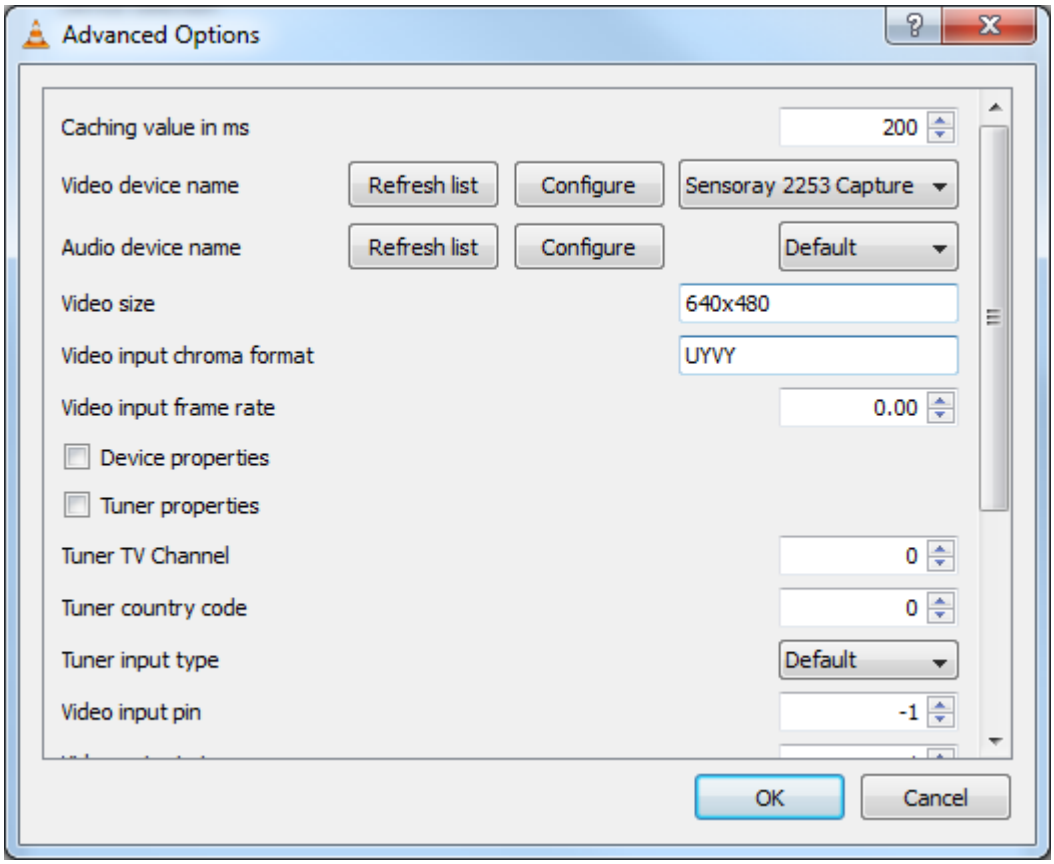
Q1) How do you view the stream(s) in the VideoLan (VLC) media player?

A1) Perform the following steps:

- Open Media->Capture Device



- Make sure Capture mode is set to DirectShow.
- Click on “Advanced options”.
- Change Video device name to “Sensoray 2253 Capture A”.
- In video size, type 640x480.
- In input chroma format, type UYVY (uppercase required) for uncompressed preview stream. Type H.264 for H.264 streaming (if saving the stream to a file).



- Click Ok (See image below for settings).
- Press Play.

Q2) Why does not the recorded H.264 file play back in Windows media player?

A2) Windows XP do not necessarily have the H.264 decoders to decode the stream. The recorded stream can be played back with WMP under Windows 7. If using Windows XP and recording H.264, the following options are available:

1. Download external H.264 codecs (not supported by Sensoray).
2. Use VLC player or Quick Time to play back the file.

Q3) How can I access the raw uncompressed data?

A3) This involves creating a callback function and using the DirectShow sample grabber. This can be done quite easily with DirectShow. If this is required from the DLL level using the Sensoray SDK, please contact support to request this feature in a future SDK. The DLL source code is also available and could be modified to support this.

MSDN documentation for using the Sample Grabber:

http://msdn.microsoft.com/en-us/library/dd407288(VS.85).aspx

Q4) How do I access the on board GPIO? There is a 2253 GPIO device in the device manager.

A4) GPIO is currently not available in the Windows SDK. It will be added in a future release.

Appendix A: Custom Preview Window example

If your application requires a custom preview window and not just the default pop-up window, the code below may be useful. It is not required, but provides an example of customizing the C++ demo application.

VidWindow.h

```
#pragma once


// CVidWindow
class CVidWindow : public CWnd
{
	DECLARE_DYNAMIC(CVidWindow)

public:
	CVidWindow();
	virtual ~CVidWindow();
	void CVidWindow::OnPaint() ;
	BOOL CVidWindow::OnEraseBkgnd(CDC *pDC) ;
protected:
	DECLARE_MESSAGE_MAP()
};
```

VidWindow.cpp

```
// VidWindow.cpp : implementation file
//

#include "stdafx.h"
#include "Demo.h"
#include "VidWindow.h"
#include "mid2253func.h"

// CVidWindow

IMPLEMENT_DYNAMIC(CVidWindow, CWnd)
```

```
CVidWindow::CVidWindow()
{
}

CVidWindow::~CVidWindow()
{
}

BEGIN_MESSAGE_MAP(CVidWindow, CWnd)
      ON_WM_PAINT()
      ON_WM_ERASEBKGND()
END_MESSAGE_MAP()

// CVidWindow message handlers
void CVidWindow::OnPaint()
{
      {
            CPaintDC dc(this);
            CWnd::OnPaint();
            // repaint the video
            S2253_RepaintWindow(dc.GetSafeHdc(), 0, 0);
      }
}

BOOL CVidWindow::OnEraseBkgnd(CDC *pDC)
{
      // return NON-ZERO,  do not erase this window or video may flicker
      return TRUE;
}
```

DemoDlg.cpp and DemoDlg.h changes

Include the header file "`vidwindow.h`". In the `OnInitDialog()` function, add the function below to create the window. "rect" is the defined window rectangle to create.

```
m_vidWin.Create(_T("STATIC"), NULL, WS_VISIBLE | WS_CHILD |
WS_CLIPSIBLINGS | WS_CLIPCHILDREN, rect, this, 9992);
```

`m_vidWin` is of type `CVidWindow`, defined in `DemoDlg.h` as a protected variable.

To set the DLL to render into the window above, call `S2253_SetStreamWindow` with an `hwnd` argument of "`m_vidWin.m_hWnd`".

Appendix B: Extended OSD example source code

Structures

```
#define MID2253_MAX_OSDTEXT 80


typedef struct mid2253_osd_text
{
   int osdOn;       //OSD on if != 0, 1=8x14 font, 2=16x16 font
   int osdChan;     // osd channel to update.  osdChan=0 for stream A, osdChan=1 for stream B, osdChan=2 for output
   int transparent; //transparent OSD if !=0, 1=100%, 2=50%, 3=25%
   int positionTop; //see xoff, yoff below
   int ddmm;     //date format 0=mm-dd 1=dd-mm 2=mmm-dd, 3=dd-mmm- 4=mmm dd, 5=ddmmm, 6=dd.mm.
   int year2;    // year display mode (year2 = 1 means 2 digits, year2=0 means 4 digits)
   int fraction; // whether to display fraction of seconds
   unsigned char line[MID2253_MAX_OSDTEXT]; // ascii string of text (null terminated)
   int xoffset; //x offset: if positionTop=1, relative to top. if positionTop=0, relative to bottom
   int yoffset; //y offset: if positionTop=1, relative to top. if positionTop=0, relative to bottom
} MID2253_OSD_TEXT;


#define MID2253_MAX_OSDLONGTEXT 160


typedef struct mid2253_osd_longtext
{
   int osdOn;       //OSD on if != 0, 1=8x14 font, 2=16x16 font
```

```
   int osdChan;     // osd channel to update.  osdChan=0 for stream A, osdChan=1 for stream B, osdChan=2 for output
   int transparent; //transparent OSD if !=0, 1=100%, 2=50%, 3=25%
   int positionTop; //see xoff, yoff below
   int ddmm;     //date format 0=mm-dd 1=dd-mm 2=mmm-dd, 3=dd-mmm- 4=mmm dd, 5=ddmmm, 6=dd.mm.
   int year2;    // year display mode (year2 = 1 means 2 digits, year2=0 means 4 digits)
   int fraction; // whether to display fraction of seconds
   int xoffset; //x offset: if positionTop=1, relative to top. if positionTop=0, relative to bottom
   int yoffset; //y offset: if positionTop=1, relative to top. if positionTop=0, relative to bottom
   unsigned char line[MID2253_MAX_OSDLONGTEXT]; // ascii string of text (null terminated)
} MID2253_OSD_LONGTEXT;


typedef struct mid2253_osd_styledtext
{
        int osdOn;              // OSD on if != 0
        int osdChan;   // osd channel to update.  osdChan=0 for stream A, osdChan=1 for stream B, osdChan=2 for output
        int id;                 // region id: 0..7
        int xoffset;      // x offset: relative to left
        int yoffset;      // y offset: relative to top
        LPSTR font;     // font name
        int size;       // point size of text
        int style;              // bit[0]: bold, bit[1]: italic, bit[2]: outline, bit[3]: underline, bit[4]: shadow
        int outline;      // outline style: 0=transparent, 1..7=shaded, 8=black
        int background; // background style: 0=transparent, 1..7=shaded, 8=black
```

```
        int color;      // RGB888 color for text (only used for osdChan=2)
        LPSTR line;     // pointer to UTF8 text (null terminated) (pointer may be NULL, which will update xoffset and yoffset only)
} MID2253_OSD_STYLEDTEXT;
```

Example of Styled OSD usage

(adapted from OsdExtended.cpp in demo program)

```
MID2253_OSD_STYLEDTEXT osd;
MID2253_OSD_DATA osddata;
TCHAR text[160];
char font[260];
char windir[260];
LPTSTR sOsd;
unsigned char osd_line[160];
GetWindowsDirectoryA(windir, 260);
memset(&osd, 0, sizeof(osd));
osd.osdOn = 1;
osd.osdChan = 0;  // Stream A
osd.id = 0;  // First OSD Window
osd.xoffset = 50;
osd.yoffset = 50;
_snprintf(font, 260, "%s\\fonts\\arial.ttf", windir);
osd.font = (LPSTR) font;
osd.size = 30; // 30 point font // point size of text
osd.style = 0;
```

```
osd.outline = 0;          // outline style: 0=transparent
osd.background = 0;  // background style: 0=transparent
osd.color = 0; // not applicable for Stream A
wsprintf(text, L”2253 Caption Test”);
sOsd =text;
WideCharToMultiByte(CP_UTF8, 0, sOsd, -1, (LPSTR) osd_line, sizeof(osd_line), NULL, NULL);
osd.line = (LPSTR) osd_line;
osddata.osdstyledtext = osd;
S2253_SetOsd(MID2253_OSDTYPE_STYLEDTEXT, &osddata, 0, 0);
```

Revision history

Version	Notes
1.0.0, August 2010	Initial release.