USB Audio/Video Codec Model 2263/3364/2231

Windows/Linux/Mac Software Manual

Ver. 1.1.26 | May 2023

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....5

RELEASE NOTES....6
- Version 1.1.26 (May *2023*)....6
- Version 1.1.24 (December *2022*)....6
- Version 1.1.23 (April *2022*)....6
- Version 1.1.22 (March *2022*)....6
- Version 1.1.21 (February *2022*)....6
- Version 1.1.20 (July *2021*)....6
- Version 1.1.19 (*May 2021*)....6
- Version 1.1.18 (*March 2021*)....7
- Version 1.1.17 (December 2020)....7
- Version 1.1.16 (October 2020)....7
- Version 1.1.15.1 (August 2020)....7
- Version 1.1.14.1 (March 2020)....7
- Version 1.1.13.2 (February 2020)....7
- Version 1.1.13.1 (June 2019)....7
- Version 1.1.11 (January 2019)....7
- Version 1.1.10 (December 2018)....8
- Version 1.1.9 (November 2018)....8
- Version 1.1.8 (October 2018)....8
- Version 1.1.7 (July 2018)....8
- Version 1.1.6 (June 2018)....8
- Version 1.1.5 (April 2018)....8
- Version 1.1.4 (August 2017)....8
- Version 1.1.3 (July 2017)....8
- Version 1.1.2 (April 2017)....8
- Version 1.1.1 (May 2016)....8
- Version 1.1.0 (Aug 2015)....8
- Version 1.0.5 (Jan 2015)....9
- Version 1.0.4 (Oct 2014)....9
- Version 1.0 (Nov 2013)....9

WINDOWS INSTALLATION....10

RE-DISTRIBUTION....10

LINUX INSTALLATION....12

MAC OS X INSTALLATION....13

BASIC OPERATION....14
- Video Capture....14
- Firmware Update....15
- GPIO....15
- Demo application....16
  - Windows....16
    - Board Index....17
  - Linux....17

Mac....17
DirectShow API....17
Video4Linux2 API....18
QTKit API....18
SDK REFERENCE....19
Initialization/Cleanup/Enumeration Functions....19
S2263_Open....19
S2263_Close....19
S2263_GetNumDevices....19
S2263_SetStreamWindow....20
S2263_SetStreamWindowPosition....20
S2263_RepaintWindow....20
S2263_GetSerialNumber....21
S2263_GetFirmwareVersion....21
S2263_FirmwareUpdate....21
S2263_GetAPIVersion....21
Stream Control Functions....22
S2263_StartRecord....22
S2263_StartPreview....22
S2263_StartDecodedPreview....22
S2263_StartDecodedPreview....23
S2263_EnableSnapshot....23
S2263_StartSnapshot....23
S2263_StopStream....24
S2263_PauseStream....24
S2263_ResumeStream....24
S2263_StopDecoded....25
S2263_GetSample....25
S2263_StartCallback....26
S2263_RegisterCallback....26
Mode Control Functions....27
S2263_SetVidSys....27
S2263_GetVidSys....27
S2263_SetInput....27
S2263_GetInput....27
S2263_GetNumInputs....28
S2263_GetVideoInfo....28
S2263_GetVideoLock....29
S2263_GetAnalogHDStandard....29
S2263_SetLevel....29
S2263_GetLevel....30
S2263_SetImageSize....30
S2263_GetImageSize....31
S2263_SetRecordMode....31
S2263_GetRecordMode....31
S2263_SetBitrate....32
S2263_GetBitrate....32
S2263_SetRateMode....33
S2263_GetRateMode....33
S2263_SetFixedQP....33

S2263_GetFixedQP....34
S2263_SetGopSize....34
S2263_GetGopSize....35
S2263_SetInterlaceMode....35
S2263_GetInterlaceMode....35
S2263_SetFrameSkip....36
S2263_GetFrameSkip....36
S2263_SetAspectRatio....36
S2263_GetAspectRatio....37
S2263_SetFlip....37
S2263_GetFlip....38
S2263_SetCrop....38
S2263_GetCrop....38
S2263_SetDeintMode....39
S2263_GetDeintMode....40
S2263_SetMP4FastStart....40
S2263_SetAudioBitrate....41
S2263_GetAudioBitrate....41
S2263_SetAudioInput....42
S2263_GetAudioInput....42
S2263_SetAudioGain....42
S2263_GetAudioGain....43
S2263_SetRemoveMsg....43
S2263_TestDeviceRemoval....43
Output Functions....44
S2263_SetOutput....44
S2263_GetOutput....44
S2263_SetOutputStd....44
S2263_GetOutputStd....45
S2263_SetOutputMode....45
S2263_GetOutputMode....46
Overlay Functions....46
S2263_OverlayText....46
S2263_OverlayStyledText....47
S2263_OverlayFile....48
S2263_OverlayData....49
S2263_OverlayMove....50
S2263_OverlayShow....50
S2263_SetFrameCount....51
S2263_GetFrameCount....51
3364 Specific Functions....51
S3364_Is3364Board....51
S3364_GetFirmwareVersionUSB3....52
S3364_GetUsbSpeed....52
S3364_GetBoardType....52
S3364_StartFullSizePreview....52
S3364_StopFullSizePreview....53
S3364_FullSizePreviewSample....53
S3364_StartCallback....53
S3364_RegisterCallback....54
S3364_StopStream....54

S3364_StartSnapshot....54
S3364_FullSizePreviewSample....54
S3364_SetUserData....55
S3364_GetUserData....55
S2263_GetFullDevicePath....56
S2263_GetFullDeviceInstance....56
REVISION HISTORY....57

Operating System Support

The Windows SDK supports Windows XP, Vista, 7, 8, 10 (32/64 bit).

The Linux SDK supports most Linux distributions with UVC driver (32/64 bit).

The Mac OS X SDK supports most Mac OS X releases with UVC driver and QTKit support (32/64 bit).

Release Notes

Version 1.1.26 (May 2023)

- Add support for Sensoray Model 2271 (DVI/SDI/VGA/Composite capture) See enum MID2263_VIDINPUT in mid2263types.h for updated video input selections.

Version 1.1.24 (December 2022)

- Adds Deinterlacer functions for SD video on a single stream: S2263_Set/GetDeintMode (Requires firmware 3793)

Version 1.1.23 (April 2022)

- Adds MID2263_RECMODE_H264 for capturing H.264 Elementary Stream.

- Firmware 3776 includes work-around for first IDR-frame being dropped.

Version 1.1.22 (March 2022)

- Adds S2263_SetFrameCount and S2263_GetFrameCount functions. The definitions for OVERLAY_TEXT_SECONDS_* have changed to match the firmware – user applications that use these will need to be recompiled. Add definitions OVERLAY_TEXT_HOURS_MINUTES +2 more with AM/PM indicators. Firmware must be updated to 3763.

- Add OSD dialog to Win32 demo program.

Version 1.1.21 (February 2022)

- Fix re-scaling and re-paint of decoded preview stream. Adds 720x405 preview size.

Version 1.1.20 (July 2021)

- Fix Windows enumeration (detection) issue in some cases on latest Windows versions.

Version 1.1.19 (May 2021)

- Fix issue with fast overlay upload not activating in some cases.

Version 1.1.18 (March 2021)

- Add functions: S2263_PauseStream, S2263_ResumeStream.
- Adds support for firmware 3636 with faster overlay upload using bulk endpoint.

Version 1.1.17 (December 2020)

- Bug fix for full size decoded preview when other video devices plugged in concurrently.

Version 1.1.16 (October 2020)

- Added full scale decoded HD preview for Windows 10. This feature is not available on Win7 or earlier. See S2263_StopDecoded, S2263_StartDecodedPreview, S2263_StartRecordDecodedPreview in this manual for details.

Version 1.1.15.1 (August 2020)

- Streaming fix for VLC. Streaming MTU size configurable. Streaming and record at same time fixed in demo application. Note: no changes to the middleware, mid2263.dll. All changes relate to demo application only.

Version 1.1.14.1 (March 2020)

- HD preview fix for 1080p30 + record (Model 3364 only)

Version 1.1.13.2 (February 2020)

- Add missing functions: S2263_SetAspectRatio, S2263_GetAspectRatio, S2263_SetFlip, S2263_GetFlip, S2263_SetCrop, S2263_GetCrop

Version 1.1.13.1 (June 2019)

- S2263_OverlayData fix for stream index 2 (Display Output)
- Fixed some functions that were missing the strmidx parameter.

Version 1.1.11 (January 2019)

- 3364 Uncompressed callback fix for S3364_RegisterCallback.

Version 1.1.10 (December 2018)

- Output idle/passthru fixes

Version 1.1.9 (November 2018)

- Demo cleanup. Preview stream stopped when window closed

Version 1.1.8 (October 2018)

- Video status fix for AHD standard
- Mic/Line input selection fix for 3364

Version 1.1.7 (July 2018)

- Stop button fix. 2231 identification and support

Version 1.1.6 (June 2018)

- Added function: S2263_GetAnalogHDStandard

Version 1.1.5 (April 2018)

- Support for Model 2231 AHD board.

Version 1.1.4 (August 2017)

- Support for Rev. C 3364 boards and functional updates.

Version 1.1.3 (July 2017)

- Minor updates

Version 1.1.2 (April 2017)

- Full 3364 support for Rev. B+ boards

Version 1.1.1 (May 2016)

- Added output functions.
- Initial 3364 Support

Version 1.1.0 (Aug 2015)

- Recording now supported on both streams – some functions take an additional strmidx parameter.

- Added functions: S2263_SetFrameSkip, S2263_GetFrameSkip
- Added functions: S2263_SetFixedQP, S2263_GetFixedQP and MID2263_RATEMODE_FIXEDQP
- Added function: S2263_GetNumInputs (for 3364 SDI)

Version 1.0.5 (Jan 2015)

- Overlay functions

Version 1.0.4 (Oct 2014)

- Preview snapshots
- Device removal detection

Version 1.0 (Nov 2013)

- Initial 2263 release.
- H264 Video encoder
- Raw video capture
- Audio Preview

Windows 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 2263 to the USB port.

Setup is performed as follows.

1) Run `setup.exe` file.
2) DLLs are installed automatically. A local copy of the DLLs are installed in the program files directory for the 2263 (typically, `C:\Program Files\Sensoray\2263\API\x32` and `C:\Program Files\Sensoray\2263\API\x64`).
3) Plug in the 2263 device.
4) On Windows, 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. Please refer to the MSDN documents online for more details on these tools if in doubt. 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\2263 in this case)

The 2263 is a UVC (USB Video Class) device, and does not require a driver to be installed from the SDK, instead uses the usbvideo.sys driver provided by the OS.

DLL(s) and ActiveX:

API\mid2263.dll

API\mp4mux.ax (register with regsvr32)

API\tsdemux.ax (register with regsvr32)

API\sraywrite.ax (register with regsvr32)

The mid2263.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 mp4mux.ax, tsdemux.ax, sraywrite.ax should be registered with regsvr32 after installation.

Linux 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 2263 to the USB port.

Setup is performed as follows.

1) Run `tar xjf sdk-2263-linux-N.NN.tar.bz2`
2) Run cd `sdk-2263-linux`
3) Run `make`
4) Plug in the 2263 device.

A python demo and API wrapper (S2263.py) is provided to get to test the features of the device.

1) Run `./demo.py`

Mac OS X 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 2263 to the USB port.

Setup is performed as follows.

1) Open `sdk-2263-osx-N.NN.zip`
2) Drag 2263Demo to the Applications folder.
3) Drag 2263DemoSource to folder of your choice.
4) Plug in the 2263 device.

To run the demo and test the features of the device, run the 2263Demo application in the Applications folder.

Basic operation

Video Capture

A 2263 device has 2 available streams, while 3364 devices have 3. The first stream is a full-resolution compressed stream with audio, suitable for streaming or storage. The compressed stream may be Motion-JPEG, or H.264 video elementary stream or with the video and AAC audio in transport stream. The transport stream may optionally be remuxed into a MP4 file.

The second stream is scaled-down uncompressed video, suitable for preview. Uncompressed video is captured in the format YUV422 (packed UYVY).

When using model 3364 device over USB3, a third stream is available for full-resolution uncompressed video, suitable for preview, in the same UYVY format.

All capture streams record video from a single source, and each capture stream can be started, stopped, and configured independently. Some options cannot be configured independently, such as brightness, hue, contrast, saturation.

On Windows, the device shows up as an Imaging Device in Windows Device Manager: Sensoray 2263. The DirectShow capture name for the device is “Sensoray 2263” with each stream available on the output pins. If multiple 2263 devices are plugged in, then multiple devices will be present. The AVStream device can be used with applications that support the DirectShow API or the Sensoray SDK. The Sensoray 2263 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.

On Linux, the device appears as a Video4Linux2 device node (/dev/videoN) and can be used with most Video4Linux2 applications. There are actually two video device nodes created: one for the scaled-down uncompressed preview stream, and one for the full-size compressed capture stream. The driver used for the video device is called “uvcvideo” and the card type will appear as “Sensoray Model 2263”. The audio preview is accessible through the ALSA API, provided by the “snd_usb_audio” driver, accessible through device nodes in the /dev/snd directory. The Sensoray 2263 SDK provides a shared library wrapper around the Video4Linux2 and ALSA API's to facilitate operation without knowledge of Video4Linux2 or ALSA programming. The shared library wrapper also allows easy porting to Python, C++ and other programming

environments. A Python demonstration program is provided to get acquainted with the device functionality.

On Mac, the device appears as a UVC device accessible through the QTKit API.

The Sensoray 2263 SDK provides a shared library wrapper around the QTKit API to facilitate operation without knowledge of QTKit and Objective C programming. The shared library wrapper also allows easy porting to C, C++ and other programming environments.

Firmware Update

The device features a flash memory that contains a firmware used to operate the hardware on the device. This firmware can updated in future SDK releases to fix problems or add new features. When selecting the firmware update option in the demo or calling the SDK function, the device will reconnect as a USB mass storage device with the name Update2263. (If the AutoPlay menu appears, choose the “Open folder to view files option”). In this mode, a new firmware file may be copied to the Update2263 folder. While the file is being written to the flash memory, the red light will blink. Do not uplug the device while the red light is blinking. After the update is complete, the device will close the Update2263 folder and reconnect as a UVC device. To cancel the update mode without updating firmware, right click the Update2263 removable device in Computer and click Eject.

In the unlikely event that a firmware update was interrupted in a way that prevents the device from operating, an original firmware mode is available. To enter original firmware mode, hold down the small button through the hole near the USB while connecting the USB cable. The RED light will illuminate, indicating the original firmware has entered firmware update mode and a new firmware image may be applied. After the procedure is complete, in order to switch to the newly updated firmware, the power needs to be cycled – disconnect and reconnect the USB cable.

GPIO

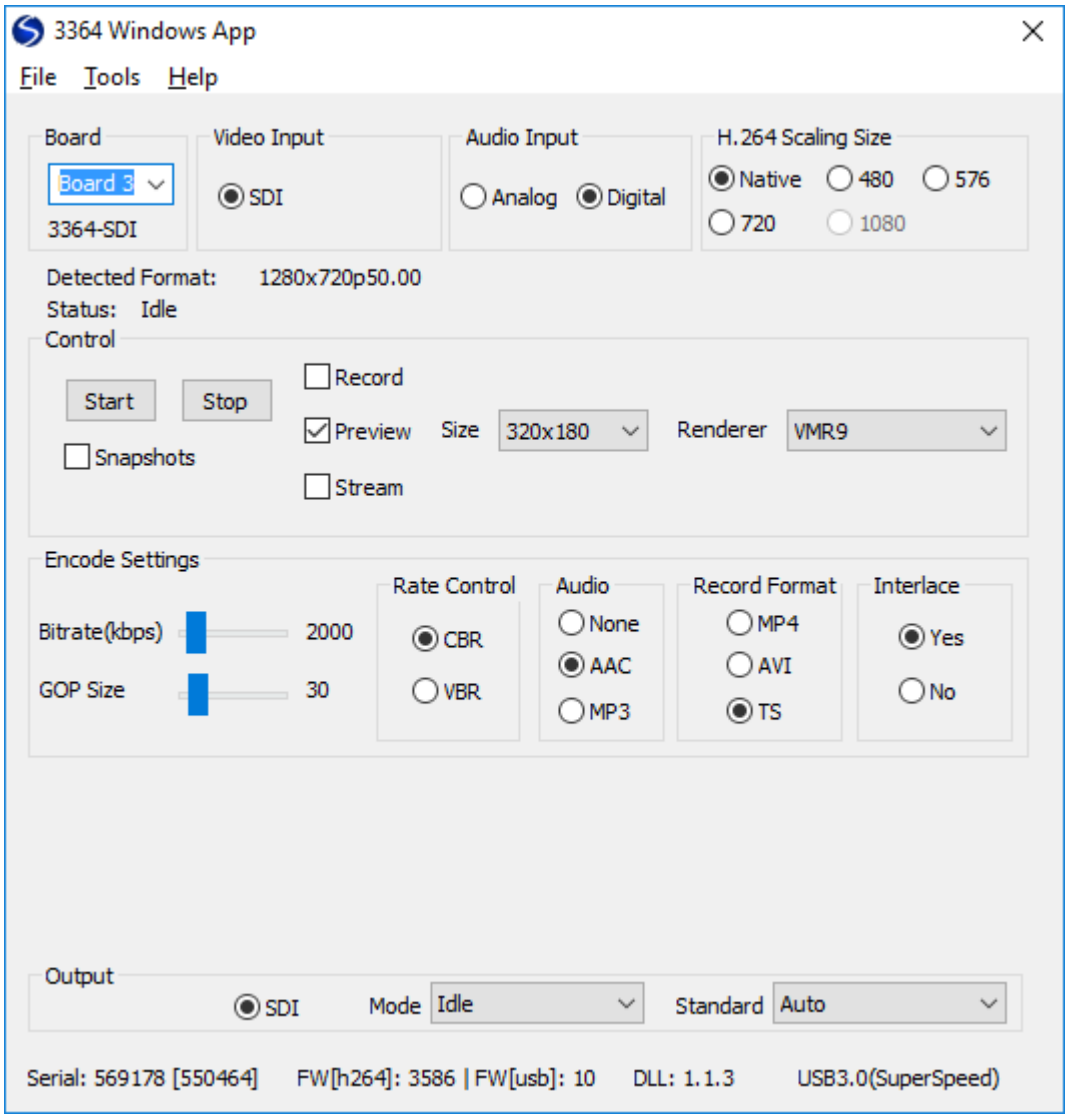
GPIO support will be offered in later revisions of the SDK.

Demo application

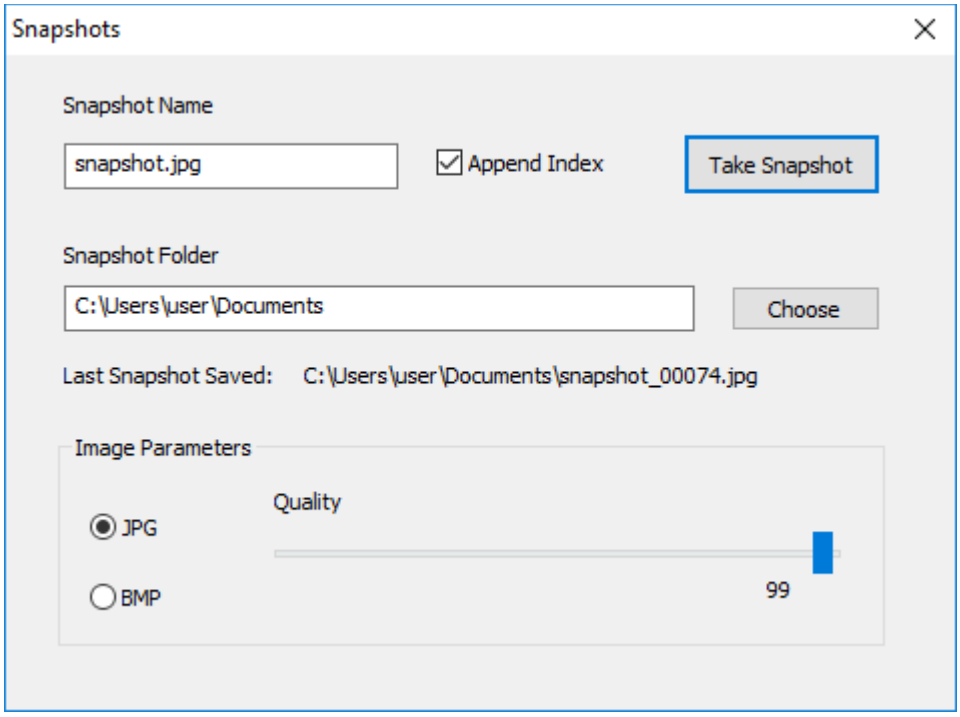
Windows

On Windows, press Start button, select Programs -> Sensoray -> 2263. Click on 2263/3364 Demo.

The Windows application allows the user to demo the SDK capabilities such as record, preview and streaming. It also demonstrates full size snapshots for the 3364 device. To start snapshots on the 3364, check the snapshots checkbox under the “Start” button. Full size snapshots require a USB3.0 connection to the 3364.



Snapshots may be taken at any time. It operates independently of recording, streaming or preview. The snapshot dialog is shown below:



Board Index

The board index for the 3364 is set by the on-board switch SW2. The default setting is board 0 with both switches in the down position. If using multiple 3364's, change the switch to another setting. The index is binary coded from 0 to 3. The board index for the 2263 will be determined by whatever board is first enumerated by USB. They can be differentiated via the serial number. Using two or more 3364s with the same index number will result in only one board being seen by the SDK.

Linux

On Linux, open a Terminal window, "cd" to the folder where the SDK was installed, and then run "./demo.py".

Mac

On Mac, open Applications folder, and click Sensoray 2263 Demo.

DirectShow API

The Windows UVC driver supports the DirectShow API. DirectShow is well documented at MSDN (http://msdn.microsoft.com/en-us/default.aspx). Most of

the device features are implemented as UVC extension unit controls, and the SDK DLL provides an API for using those features.

Video4Linux2 API

The Linux UVC driver supports the Video4Linux2 API. Video4Linux2 is well documented here (http://linuxtv.org/downloads/v4l-dvb-apis/). Most of the device features are implemented as UVC extension unit controls, and the SDK shared library provides an API for using those features.

QTKit API

The Mac UVC driver supports the QTKit Framework. QTKit Framework is well documented at Apple web site (https://developer.apple.com/library/mac/documentation/quicktime/reference/qtcocoaobjckit/_index.html). Most of the device features are implemented as UVC extension unit controls, and the SDK shared library provides an API for using those features.

SDK Reference

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

```
#define MID2263_API extern "C" __declspec(dllimport)
```

For example,

```
MID2263_API int __stdcall S2263_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 `mid2263types.h` for error codes list).

Initialization/Cleanup/Enumeration Functions

S2263_Open

```
MID2263_API int __stdcall S2263_Open (
      int               board_index);
```

Must be called before any other API functions are called. If called with a -1 parameter, all 2263 boards in the system will be available after the call.

`board_index`
Zero based index of a 2263 board (or -1 for all boards).

S2263_Close

```
MID2263_API int __stdcall S2263_Close (
      int               board_index);
```

Must be called before application terminates for proper clean-up of the SDK and SDK objects when SDK opened with S2263_Open.

`board_index`
Zero based index of a 2263 board (or -1 for all boards).

S2263_GetNumDevices

```
MID2263_API int __stdcall S2263_GetNumDevices (
      int               *NumDevices);
```

Retrieves the number of devices in the system. Only valid after S2263_Open is called.

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

S2263_SetStreamWindow

```
MID2263_API int __stdcall S2263_SetStreamWindow (
      HWND              hwnd,
      int               devid);
```

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 `S2263_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).

S2263_SetStreamWindowPosition

```
MID2263_API int __stdcall S2263_SetStreamWindowPosition (
      RECT              rcDst,
      int               devid);
```

Used to set the position of the stream window.

`rcDst`
Coordinates of the window position.

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

S2263_RepaintWindow

```
MID2263_API int __stdcall S2263_RepaintWindow (
      HDC               hdc,
      int               devid);
```

Used only if `S2263_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).

S2263_GetSerialNumber

```
MID2263_API int __stdcall S2263_GetSerialNumber (
      int               *serial_number,
      int               devid);
```

Retrieves the serial number from the 2263. Each 2263 has a unique serial number.

`serial_number`
a pointer to the variable receiving the serial number of device.

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

S2263_GetFirmwareVersion

```
MID2263_API int __stdcall S2263_GetFirmwareVersion (
      int               *firmware_version,
      int               devid);
```

Retrieves the firmware version from the 2263.

`firmware_version`
a pointer to the variable receiving the firmware version of device.

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

S2263_FirmwareUpdate

```
MID2263_API int __stdcall S2263_FirmwareUpdate (
      int               devid);
```

Put the device in a firmware update mode.

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

S2263_GetAPIVersion

```
MID2263_API int __stdcall S2263_GetAPIVersion ();
```

This returns the current version of the 2263 DLL or shared library. The version number can extracted from the result as follows:

```
rc = S2263_GetAPIVersion();
major_version = (rc >> 16) & 0xff;
minor_version = (rc >> 8) & 0xff;
release_version = rc & 0xff;
```

Stream Control Functions

S2263_StartRecord

```
MID2263_API int __stdcall S2263_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).

S2263_StartPreview

```
MID2263_API int __stdcall S2263_StartPreview (
      int               devid,
      int               strmidx);
```

Starts video stream and displays video in a pop-up window or in a window handle set by S2263_SetStreamWindow.

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

`strmidx`
stream index (0 or 1).

S2263_StartDecodedPreview

```
MID2263_API int __stdcall S2263_StartDecodedPreview (
      int               devid,
      int               strmidx);
```

Starts video stream and displays video in a pop-up window or window handle set previously with S2263_SetStreamWindow. Decodes H.264 stream sent from the hardware and displays at the encoded resolution. Available on Windows 10 or later only. Stop with S2263_StopDecoded.

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

`strmidx`
must be 0. Stream 1 not supported.

S2263_StartDecodedPreview

```
MID2263_API int __stdcall S2263_StartRecordDecodedPreview (
      const void        *fileName,
      BOOL              bUnicode,
      int               devid,
      int               strmidx);
```

Starts video stream and displays video in a pop-up window or window handle set previously with S2263_SetStreamWindow while recording stream to file. Decodes H.264 stream sent from the hardware and displays at the encoded resolution. Available on Windows 10 or later only. Stop with S2263_StopDecoded.

`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`
must be 0. Stream 1 not supported.

S2263_EnableSnapshot

```
MID2263_API int __stdcall S2263_EnableSnapshot (
      BOOL              bOn,
      int               devid,
      int               strmidx);
```

Enables or disables snapshots (using `S2263_GetSample`) for preview or record streams. Disabling snapshots decreases CPU usage. Snapshots are disabled by default. Note: this function is only available on Windows SDK, and only makes sense for uncompressed or MJPEG streams.

`bOn`
`TRUE` - enables `S2263_GetSample` function, `FALSE` - disables it.

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

`strmidx`
stream index (0 or 1).

S2263_StartSnapshot

```
MID2263_API int __stdcall S2263_StartSnapshot (
```

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

Starts a snapshot stream. Device will stream and samples will be buffered until S2263_GetSample is called. Stream will not be previewed or recorded. Stop stream with S2263_StopStream. Snapshot stream is independent from preview or record stream. Snapshots can still be obtained with preview or record by using S2263_EnableSnapshot. Note: this function is only available on Windows SDK, and only makes sense for uncompressed or MJPEG streams.

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

`strmidx`
stream index (0 or 1).

S2263_StopStream

```
MID2263_API int __stdcall S2263_StopStream (
    int             devid,
    int             strmidx);
```

Stops a stream (callback, record, preview, snapshot).

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

`strmidx`
stream index (0 or 1).

S2263_PauseStream

```
MID2263_API int __stdcall S2263_PauseStream (
    int             devid,
    int             strmidx);
```

Pause a record stream (no effect on preview). Use S2263_ResumeStream to resume. This effectively cause a jump/cut in the recording from the time of the pause to the time of the resume.

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

`strmidx`
stream index (0 or 1).

S2263_ResumeStream

```
MID2263_API int __stdcall S2263_ResumeStream (
    int             devid,
    int             strmidx);
```

Resumes a paused recording stream (no effect on preview). Use S2263_PauseStream to pause.

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

`strmidx`
stream index (0 or 1).

S2263_StopDecoded

```
MID2263_API int __stdcall S2263_StopDecoded (
      int               devid,
      int               strmidx);
```

Stops a stream started with S2263_StartDecodedPreview or S2263_StartRecordDecodedPreview.

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

`strmidx`
stream index (0).

S2263_GetSample

```
MID2263_API int __stdcall S2263_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 `S2263_StartSnapshot, S2263_StartPreview or S2263_StartRecord`. `S2263_GetSample` is used only for uncompressed (UYVY) streams.
Note: this function is only available on Windows SDK.

`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).

S2263_StartCallback

```
MID2263_API int __stdcall S2263_StartCallback (
      int               devid,
      int               strmidx);
```

Start callback allows the user to capture data to a callback function that was previously registered with S2263_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 S2263_RegisterCallback for details about the callback routine.

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

`strmidx`
stream index (0 or 1).

S2263_RegisterCallback

```
MID2263_API int __stdcall S2263_StartCallback (
      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).

Mode Control Functions

S2263_SetVidSys

```
MID2263_API int __stdcall S2263_SetVidSys (
      MID2263_VIDSYS    vidsys,
      int               devid);
```

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

`vidsys`
video system enumerated type (see `mid2263types.h`).

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

S2263_GetVidSys

```
MID2263_API int __stdcall S2263_GetVidSys (
      MID2263_VIDSYS    *vidsys,
      int               devid);
```

Gets the input video system (NTSC, PAL).

`vidsys`
pointer to video system enumerated type (see `mid2263types.h`).

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

S2263_SetInput

```
MID2263_API int __stdcall S2263_SetInput (
      MID2263_VIDINPUT  input,
      int               devid);
```

Sets the video input source (DVI, Composite, Component).

`input`
video input enumerated type (see `mid2263types.h`).

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

S2263_GetInput

```
MID2263_API int __stdcall S2263_GetInput (
      MID2263_VIDINPUT  *input,
      int               devid);
```

Gets the input input source.

`input`
pointer to video input enumerated type (see `mid2263types.h`).

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

S2263_GetNumInputs

```
MID2263_API int __stdcall S2263_GetNumInputs (
      int               *inputs,
      int               devid);
```

Gets the number of available video input sources. For the model 3364 with SDI input, the number of inputs will be 1. For the model 2231 with AHD inputs, the number of inputs will be 2.

`inputs`
pointer to int receiving the number of video inputs

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

S2263_GetVideoInfo

```
MID2263_API int __stdcall S2263_GetVideoInfo (
      int               *width,
      int               *height,
      int               *framerate,
      int               *interlaced,
      int               devid);
```

Gets information about the currently selected video source. If there is no video lock when the function is called, the return value will be `-1`. When interlaced returns 1, the `framerate` parameter is actually fields instead of frames.

`width`
pointer to variable receiving width in pixels.

`height`
pointer to variable receiving height in pixels.

`framerate`
pointer to variable receiving video frequency scaled by 1000.
When `*interlaced=0`, this is **frames** per second x1000.
When `*interlaced=1`, this is **fields** per second x1000.

`interlaced`
pointer to variable receiving boolean indicating interlaced video. This determines if the `framerate` parameter is fields or frames.

`devid`

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

S2263_GetVideoLock

```
MID2263_API int __stdcall S2263_GetVideoLock (
      int               *locked,
      int               devid);
```

Gets signal status for the currently selected video source.

`locked`
pointer to variable receiving video lock status: 0=no signal, 1=locked.

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

S2263_GetAnalogHDStandard

```
MID2263_API int __stdcall S2263_GetAnalogHDStandard (
      int               *standard,
      int               devid);
```

Gets analog HD standard for the currently selected video source, when locked. Use with analog HD board model 2231. Requires firmware 3382 or later.

`standard`
pointer to variable receiving analog HD standard: 0=CVBS, 1=AHD, 2=TVI, 3=CVI. (See MID2263_AHD_STANDARD constants)

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

S2263_SetLevel

```
MID2263_API int __stdcall S2263_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 (`MID2263_LEVEL_CONTRAST`, `MID2263_LEVEL_BRIGHTNESS`, `MID2263_LEVEL_SATURATION`, `MID2263_LEVEL_HUE`). See `mid2263types.h` for definitions.

`value`
defines the value of selected parameter.

`devid`

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

S2263_GetLevel

```
MID2263_API int __stdcall S2263_GetLevel (
      int               param,
      unsigned char     *value,
      int               devid);
```

Retrieves current brightness, contrast, saturation and hue settings.

`param`
  defines the parameter to get (`MID2263_LEVEL_CONTRAST`, `MID2263_LEVEL_BRIGHTNESS`, `MID2263_LEVEL_SATURATION`, `MID2263_LEVEL_HUE`). See see `mid2263types.h` for definitions.

`value`
  pointer to returned value of selected parameter.

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

S2263_SetImageSize

```
MID2263_API int __stdcall S2263_SetImageSize (
      int               width,
      int               height,
      int               devid,
      int               strmidx);
```

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

For strmidx=0: 720x480, 720x576, 1280x720, 1920x1080.

For strmidx=1: 720x480, 720x576, 640x480, 640x360, 512x384, 512x288, 320x240, 320x180.

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 `S2263_SetImageSize` with a call to `S2263_GetImageSize` (see below), which will return the actual values set in the firmware, and use those values in the application.

Please note that `S2263_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).

S2263_GetImageSize

```
MID2263_API int __stdcall S2263_GetImageSize (
      int               *width,
      int               *height,
      int               devid,
      int               strmidx);
```

Retrieves currently set image size (resolution). The values may not match those passed by `S2263_SetImageSize` function, in case those were invalid. If the size changes, it will change after the stream is started. Therefore it is best to call `S2263_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).

S2263_SetRecordMode

```
MID2263_API int __stdcall S2263_SetRecordMode(
      MID2263_RECMODE         recmode,
      int                     devid,
      int                     strmidx);
```

Sets the recording mode for saved files, such as ts/mp4/avi/H.264ES. Please see `mid2263types.h` for the description.

`recmode`
One of supported recording modes.

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_GetRecordMode

```
MID2263_API int __stdcall S2263_GetRecordMode(
```

```
      MID2263_RECMODE         *recmode,
      int                     devid,
      int                     strmidx);
```

Gets the curently set recording mode for saved files. Please see `mid2263types.h` for the description.

`recmode`
A pointer to the variable receiving the recording mode.

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_SetBitrate

```
MID2263_API int __stdcall S2263_SetBitrate(
      int               bitrate,
      int               devid,
      int               strmidx);
```

Sets the bitrate for H.264 encoding, in kilobits per second (kbps). Allowed range: 100-20000 kbps.

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

`bitrate`
bitrate in kbps.

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_GetBitrate

```
MID2263_API int __stdcall S2263_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). This parameter was added in SDK version 1.1.

S2263_SetRateMode

```
MID2263_API int __stdcall S2263_SetRateMode(
      MID2263_RATEMODE  ratemode,
      int               devid,
      int               strmidx);
```

Sets the rate mode for controlling image quality and compression ratio. The `S2263_RATEMODE_CBR` mode attempts a constant bit rate. The `S2263_RATEMODE_VBR`, mode uses variable bit rate. The `S2263_RATEMODE_FIXEDQP` mode uses a constant quality mode, and the bit rate depends on the complexity of the image.

`ratemode`
  one of `S2263_RATEMODE_CBR`, `S2263_RATEMODE_VBR`, `S2263_RATEMODE_FIXEDQP`

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

`strmidx`
  stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_GetRateMode

```
MID2263_API int __stdcall S2263_GetRateMode(
      MID2263_RATEMODE *ratemode,
      int               devid,
      int               strmidx);
```

Gets the current rate mode for controlling image quality and compression ratio.

`ratemode`
  a pointer to the variable receiving the rate mode.

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

`strmidx`
  stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_SetFixedQP

```
MID2263_API int __stdcall S2263_SetFixedQP(
      int               intraframeqp,
      int               interpframeqp,
      int               devid,
      int               strmidx);
```

Sets the fixed quality parameters for H.264 encoding, when the rate mode is set to FIXEDQP.

`intraframeqp`
quality parameter 0(highest)..51(lowest) for encoding I-frames

`interpframeqp`
quality parameter 0(highest)..51(lowest) for encoding P-frames

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_GetFixedQP

```
MID2263_API int __stdcall S2263_GetFixedQP(
      int               *intraframeqp,
      int               *interpframeqp,
      int               devid,
      int               strmidx);
```

Get the fixed quality parameters for H.264 encoding, when the rate mode is set to FIXEDQP.

`intraframeqp`
a pointer to the variable receiving quality parameter for I-frames.

`interpframeqp`
a pointer to the variable receiving quality parameter for P-frames.

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_SetGopSize

```
MID2263_API int __stdcall S2263_SetGopSize (
      int               gopsize,
      int               devid,
      int               strmidx);
```

Sets the GOP (group of pictures) size for compressed streams (H.264). Use 1 to 255, default 30.

`gopsize`
gop size. (Use 30 for default).

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_GetGopSize

```
MID2263_API int __stdcall S2263_GetGopSize (
      int               *gopsize,
      int               devid,
      int               strmidx);
```

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

`gopsize`
a pointer to the variable receiving the gop size.

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_SetInterlaceMode

```
MID2263_API int __stdcall S2263_SetInterlaceMode (
      int               val,
      int               devid,
      int               strmidx);
```

Sets interlacing option on compressed streams when the input source is interlaced. this option controls interlacing or deinterlacing (by dropping one field and interpolating) the compressed stream. When the input source is progressive, this option has no effect; output stream is always progressive.

`val`
0 – deinterlace mode(interpolation); 1 – interlaced mode is on.

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_GetInterlaceMode

```
MID2263_API int __stdcall S2263_GetInterlaceMode (
      int               *val,
      int               devid,
      int               strmidx);
```

Retrieves interlaced mode.

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

`devid`

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

`strmidx`
stream index (0 or 1). This parameter was added in SDK version 1.1.

S2263_SetFrameSkip

```
MID2263_API int __stdcall S2263_SetFrameSkip (
      int               skip,
      int               devid,
      int               strmidx);
```

Sets frame skip option on the specified stream.

`skip`
0 – all frames, 1 – every other frame, 2 – every third frame, etc.

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

`strmidx`
stream index (0 or 1).

S2263_GetFrameSkip

```
MID2263_API int __stdcall S2263_GetFrameSkip (
      int               *skip,
      int               devid,
      int               strmidx);
```

Retrieves frame skip.

`val`
a pointer to the variable receiving frame skip.

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

`strmidx`
stream index (0 or 1).

S2263_SetAspectRatio

```
MID2263_API int __stdcall S2263_SetAspectRatio (
      MID2263_ASPECT    aspect,
      int               devid,
      int               strmidx);
```

Sets aspect ratio option on the specified stream. This is stored in the H.264 stream metadata for video players to correctly determine the aspect ratio for displaying the video.

`aspect`

MID2263_ASPECT type:

```
MID2263_ASPECT_NONE   // square pixel
MID2263_ASPECT_4_3    // 4:3 SD
MID2263_ASPECT_16_9   // 16:9 HD
```

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

`strmidx`
stream index (0 or 1).

S2263_GetAspectRatio

```
MID2263_API int __stdcall S2263_GetAspectRatio (
    MID2263_ASPECT    *aspect,
    int               devid,
    int               strmidx);
```

Retrieves aspect ratio.

`aspect`
a pointer to the variable receiving aspect ratio.

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

`strmidx`
stream index (0 or 1).

S2263_SetFlip

```
MID2263_API int __stdcall S2263_SetFlip (
    int               flip,
    int               devid,
    int               strmidx);
```

Sets vertial or horizontal image flipping on the specified stream.

`flip`
0 – flipping is disabled
1 – flip image vertically (MID2263_FLIP_V)
2 – flip image horizontally (MID2263_FLIP_H)
3 – flip image both vertically and horizontally (effectively rotate 180 deg) (MID2263_FLIP_V + MID2263_FLIP_H)

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

`strmidx`
stream index (0 or 1).

S2263_GetFlip

```
MID2263_API int __stdcall S2263_GetFlip (
      int               *flip,
      int               devid,
      int               strmidx);
```

Retrieves image flipping mode.

`flip`
a pointer to the variable receiving flip mode.

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

`strmidx`
stream index (0 or 1).

S2263_SetCrop

```
MID2263_API int __stdcall S2263_SetCrop (
      int               left,
      int               top,
      int               width,
      int               height,
      int               devid,
      int               strmidx);
```

Sets crop/zoom window on the specified stream. The window is relative to the resolution of the input source returned by S2263_GetVideoInfo. If the crop window width and height are less than the size of the stream image size (S2263_SetImageSize), the resizer will scale up (or zoom) the stream image. For example, if the input source in 1920x1080, the crop settings of 480,270,960,540 will zoom-in 2X at the center of the input image.

`left, top, width, height`
The dimensions of the rectangle describing the portion of the input video to be cropped/zoomed.

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

`strmidx`
stream index (0 or 1).

S2263_GetCrop

```
MID2263_API int __stdcall S2263_GetCrop (
      int               *left,
      int               *top,
      int               *width,
      int               *height,
      int               devid,
      int               strmidx);
```

Retrieves crop/zoom window.

`left, top, width, height`
pointers to the variables receiving window dimensions.

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

`strmidx`
stream index (0 or 1).

S2263_SetDeintMode

```
MID2263_API int __stdcall S2263_SetDeintMode (
      int                     enable,
      int                     channel,
      int                     threshold_low,
      int                     threshold_high,
      int                     exclude_start,
      int                     exclude_count,
      int                     devid)
```

Sets the deinterlacer settings. Deinterlacer will only activate if stream resolution is 720x576 or smaller, and the video input is interlaced. (Requires firmware 3793 or later.)

`enable`
Use to 0 to disable (default), 1 to enable.

`channel`
Use to 0 for recording Stream A (default), 1 for preview Stream B.

`threshold_low`
Low threshold, default 4.

`threshold_high`
High threshold, default 20. Note: high minus low must be equal to one of 1,2,4,8,16,32,64,128.

`exclude_start`
Starting line number of data lines to exclude from deinterlace process.

`exclude_count`
Count of lines to exclude from deinterlace process.

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

S2263_GetDeintMode

```
MID2263_API int __stdcall S2263_GetDeintMode (
      int                   *enable,
      int                   *channel,
      int                   *threshold_low,
      int                   *threshold_high,
      int                   *exclude_start,
      int                   *exclude_count,
      int                   devid)
```

Gets the current deinterlacer settings.

`enable`
a pointer to the variable receiving the deinterlacer enable.

`channel`
a pointer to the variable receiving the deinterlacer channel.

`threshold_low`
a pointer to the variable receiving the threshold low value.

`threshold_high`
a pointer to the variable receiving the threshold high value.

`exclude_start`
a pointer to the variable receiving the exclude starting line.

`exclude_count`
a pointer to the variable receiving the excluded line count.

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

S2263_SetMP4FastStart

```
MID2263_API int __stdcall S2263_SetMp4FastStart (
      int             max_minutes,
      int             interval_sec,
      int             devid,
      int             strmidx);
```

Sets the fast start mode for writing MP4 files. This places the movie header at the beginning of the file, so that it will start playing more quickly when downloaded by a web browser. When allocating the space for the header in the file, the size of the allocation is estimated by the expected maximum duration of the recording. If the recording time exceeds the maximum duration, the header will be written at the end of the file instead and will not have fast start playback. The `interval_sec` parameter allows the header to be written to the file at a specific interval, allowing the file to be played as it is

recording, or to attempt to have a recoverable recording if the system crashes or loses power.

`max_minutes`
The expected maximum duration of the recording in minutes, used to calculate the size of the allocated header at the start of the file. When zero, the header will always be placed at the end of the file.

`interval_sec`
How often to write the header to the file in seconds. Note: `max_minutes` must not be zero.

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

`strmidx`
stream index (0 or 1).

S2263_SetAudioBitrate

```
MID2263_API int __stdcall S2263_SetAudioBitrate (
      MID2263_AUDIO_BITRATE   audbr,
      int                     devid);
```

Sets the audio encoding bitrate. This setting only applies to files recorded with AAC or MP3 audio. The audio encoding must be set before the stream is started. On the fly changes are not supported.

`audbr`
MID2263_AUDBR_96 (96kbps), MID2263_AUDBR_128 (128kbps), MID2263_AUDBR_192 (192kbps), MID2263_AUDBR_224 (224kbps), MID2263_AUDBR_256 (256kbps).

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

S2263_GetAudioBitrate

```
MID2263_API int __stdcall S2263_GetAudioBitrate (
      MID2263_AUDIO_BITRATE   *audbr,
      int                     devid);
```

Get current audio bitrate setting.

`audbr`
a pointer to the variable receiving the audio bitrate.

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

S2263_SetAudioInput

```
MID2263_API int __stdcall S2263_SetAudioInput (
      MID2263_AUDIO_INPUT     input,
      int                     devid)
```

Sets the audio input.

`input`
MID2263_AUDIO_ANALOG, MID2263_AUDIO_DIGITAL.

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

S2263_GetAudioInput

```
MID2263_API int __stdcall S2263_GetAudioInput (
      MID2263_AUDIO_INPUT     *input,
      int                     devid)
```

Gets the current audio input setting.

`input`
a pointer to the variable receiving the audio input.

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

S2263_SetAudioGain

```
MID2263_API int __stdcall S2263_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 (S2263_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).

S2263_GetAudioGain

```
MID2263_API int __stdcall S2263_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).

S2263_SetRemoveMsg

```
MID2263_API int __stdcall S2263_SetRemoveMsg (
      HWND        messageHwnd,
      int         removeMsg,
      int         devid)
```

Register the event removeMsg to be delivered to the window messageHwnd when the device is disconnected. When the removeMsg event is received, the application should call S2263_TestDeviceRemoval to test if the device was removed. After the device is removed, API functions will fail, and the DLL should be closed.

This function is only available in the Windows SDK.

`messageHwnd`
handle to the window

`removeMsg`
the event message to be sent to the window, typically (WM_APP + N)

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

S2263_TestDeviceRemoval

```
MID2263_API int __stdcall S2263_TestDeviceRemoval (
      int         devid)
```

Determines if the device has been removed. This should be called by the application after it receives the removeMsg event set by

S2263_SetRemoveMsg. Returns TRUE if the device has been removed. After the device is removed, API functions will fail, and the DLL should be closed.

This function is only available in the Windows SDK.

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

Output Functions

S2263_SetOutput

```
MID2263_API int __stdcall S2263_SetOutput (
      MID2263_OUTPUT    output,
      int               devid)
```

output
Use one of the following enums:

MID2263_OUTPUT_CVIDEO
MID2263_OUTPUT_DVI

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

S2263_GetOutput

```
MID2263_API int __stdcall S2263_GetOutput (
      MID2263_OUTPUT    *output,
      int               devid)
```

output
Pointer to receive the current output enum.

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

S2263_SetOutputStd

```
MID2263_API int __stdcall S2263_SetOutputStd (
      MID2263_OUTPUT_STD      std,
      int                     devid)
```

std
Use one of the following enums:

MID2263_OUTPUT_AUTO
MID2263_OUTPUT_VGA
MID2263_OUTPUT_NTSC
MID2263_OUTPUT_PAL
MID2263_OUTPUT_480P60
MID2263_OUTPUT_576P50
MID2263_OUTPUT_720P60
MID2263_OUTPUT_720P50
MID2263_OUTPUT_720P30
MID2263_OUTPUT_1080I60
MID2263_OUTPUT_1080I50
MID2263_OUTPUT_1080P30
MID2263_OUTPUT_1080P25
MID2263_OUTPUT_1080P24

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

S2263_GetOutputStd

```
MID2263_API int __stdcall S2263_GetOutputStd (
      MID2263_OUTPUT_STD      *std,
      int                     devid)
```

`std`
Pointer to receive the current output standard enum.

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

S2263_SetOutputMode

```
MID2263_API int __stdcall S2263_SetOutputMode (
      MID2263_OUTPUT_MODE     mode,
      int                     devid)
```

`mode`
Use one of the following enums:

MID2263_OUTPUT_IDLE	(black screen)
MID2263_OUTPUT_PASSTHRU	(primary stream)
MID2263_OUTPUT_PASSTHRU_SD	(secondary stream)

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

S2263_GetOutputMode

```
MID2263_API int __stdcall S2263_GetOutputMode (
      MID2263_OUTPUT_MODE     *mode,
      int                     devid)
```

`mode`
Pointer to receive the current output mode enum.

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

Overlay Functions

S2263_OverlayText

```
MID2263_API int __stdcall S2263_OverlayText (
      char        *text,
      unsigned int mode,
      int         x,
      int         y,
      int         devid,
      int         strmidx)
```

`text`
Pointer to ASCII character string to display. Use the following codes to display extra information in the text:

^d (date)
^t (time)
^n (newline)
^c (frame counter)

`mode`
Combine the following mode flags to adjust the display of the text:

OVERLAY_TEXT_DISABLED:	turn off text overlay
OVERLAY_TEXT_FONT_8x14:	turn on text overlay with small font
OVERLAY_TEXT_FONT_16x16:	turn on text overlay with large font
OVERLAY_TEXT_BG_SOLID:	solid black background
OVERLAY_TEXT_BG_TRANSPARENT:	no background
OVERLAY_TEXT_BG_SHADED:	50% shaded black background
OVERLAY_TEXT_BG_DARKSHADED:	75% shaded black background
OVERLAY_TEXT_DATE_MM_DD_YYYY:	date format 01-30-1999
OVERLAY_TEXT_DATE_DD_MM_YYYY:	date format 30-01-1999
OVERLAY_TEXT_DATE_MM_DD_YY:	date format 01-30-99
OVERLAY_TEXT_DATE_DD_MM_YY:	date format 30-01-99

OVERLAY_TEXT_DATE_MMM_DD_YYYY: date format Jan-30-1999
OVERLAY_TEXT_DATE_DD_MMM_YYYY: date format 30-Jan-1999
OVERLAY_TEXT_DATE_MMM_DD_YY: date format Jan-30-99
OVERLAY_TEXT_DATE_DD_MMM_YY: date format 30-Jan-99
OVERLAY_TEXT_DATE_MMM_DDc_YYYY: date format Jan 30, 1999
OVERLAY_TEXT_DATE_DDMMMYYYY: date format 30Jan1999
OVERLAY_TEXT_DATE_MMM_DDc_YY: date format Jan 30, 99
OVERLAY_TEXT_DATE_DDMMMYY: date format 30Jan99
OVERLAY_TEXT_DATE_YYYYMMDD: date format 19990130
OVERLAY_TEXT_DATE_YYYYMMMDD: date format 1999Jan30
OVERLAY_TEXT_DATE_YYMMDD: date format 990130
OVERLAY_TEXT_DATE_YYMMMDD: date format 99Jan30
OVERLAY_TEXT_DATE_YYYY_MM_DD: date format 1999-01-30
OVERLAY_TEXT_DATE_YYYY_MMM_DD: date format 1999-Jan-30
OVERLAY_TEXT_DATE_YY_MM_DD: date format 99-01-30
OVERLAY_TEXT_DATE_YY_MMM_DD: date format 99-Jan-30
OVERLAY_TEXT_SECONDS_NONE: time format HH:MM:SS
OVERLAY_TEXT_SECONDS_TENTHS: time format HH:MM:SS.0
OVERLAY_TEXT_SECONDS_HUNDREDTHS: time format HH:MM:SS.00
OVERLAY_TEXT_SECONDS_THOUSANDTHS: time format HH:MM:SS.000
OVERLAY_TEXT_HOURS_MINUTES: time format HH:MM
OVERLAY_TEXT_HOURS_MINUTES_AM_PM: time format HH:MMAM
OVERLAY_TEXT_HOURS_MINUTES_A_P: time format HH:MMa

x

Horizontal position of the upper left corner of the text overlay.

y

Vertical position of the upper left corner of the text overlay.

devid

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

strmidx

stream index (0 or 1).

S2263_OverlayStyledText

```
MID2263_API int __stdcall S2263_OverlayStyledText (
      char        *text,
      char        *font_name,
      int         font_size,
      unsigned int style,
      int         x,
      int         y,
      int         alpha,
      int         devid,
```

```
      int         strmidx,
      int         channel)
```

`text`
Pointer to ASCII or UTF8 character string to display.

`font_name`
Pointer to the name of the font to use.

`font_size`
Size of the font in points.

`style`
Combine the following style flags to adjust the appearance of the text.

OVERLAY_STYLE_BOLD:	bold text
OVERLAY_STYLE_ITALIC:	italic text
OVERLAY_STYLE_UNDERLINE:	underlined text
OVERLAY_STYLE_OUTLINE:	white text over black outline
OVERLAY_STYLE_SHADOW:	shaded shadow underneath

x

Horizontal position of the upper left corner of the text overlay.

`y`
Vertical position of the upper left corner of the text overlay.

`alpha`
Background alpha control, 0=transparent, 8=solid black.

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

`strmidx`
stream index (0 or 1).

`channel`
Which overlay channel to use: 0 to 15

S2263_OverlayFile

```
MID2263_API int __stdcall S2263_OverlayFile (
      char        *filename,
      int         x,
      int         y,
      int         alpha,
      int         devid,
      int         strmidx,
      int         channel)
```

Loads an PNG or BMP image file at the specified overlay channel. This function is not available on Windows, instead use the function S2263_OverlayData with file data loaded into a buffer.

x

Horizontal position of the upper left corner of the text overlay.

y

Vertical position of the upper left corner of the text overlay.

alpha

Image alpha control, 0=transparent, 8=solid black.

devid

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

strmidx

stream index (0 or 1).

channel

Which overlay channel to use: 0 to 15

S2263_OverlayData

```
MID2263_API int __stdcall S2263_OverlayData (
      void        *data,
      unsigned int length,
      int         x,
      int         y,
      int         alpha,
      int         devid,
      int         strmidx,
      int         channel)
```

Loads an PNG or BMP image file at the specified overlay channel. When using strmidx 2, the changes are double-buffered on the display; to make display changes visible, call S2263_OverlayData with data pointing to 4 byte string ‘updt’.

data

Pointer to PNG or BMP image data.

length

Length of the data in bytes.

x

Horizontal position of the upper left corner of the image overlay.

y

Vertical position of the upper left corner of the image overlay.

alpha

Image alpha control, 0=transparent, 8=solid, 9=use image alpha.

devid

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

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

channel
Which overlay channel to use: 0 to 15

S2263_OverlayMove

```
MID2263_API int __stdcall S2263_OverlayMove (
      int         x,
      int         y,
      int         alpha,
      int         devid,
      int         strmidx,
      int         channel)
```

Moves an existing StyledText or Image overlay to a new position, and alpha mode. This is more efficient than loading the same overlay channel at a new position.

x

Horizontal position of the upper left corner of the text overlay.

y
Vertical position of the upper left corner of the text overlay.

alpha
Image alpha control, 0=transparent, 8=solid black.

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

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

channel
Which overlay channel to use: 0 to 15

S2263_OverlayShow

```
MID2263_API int __stdcall S2263_OverlayShow (
      int         visible,
      int         devid,
      int         strmidx,
      int         channel)
```

Show or hide an existing StyledText or Image overlay.

visible
Show or hide the overlay, 0=hide, 1=show.

devid

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

`strmidx`
stream index (0 or 1).

`channel`
Which overlay channel to use: 0 to 15

S2263_SetFrameCount

```
MID2263_API int __stdcall S2263_SetFrameCount (
      int               count,
      int               devid);
```

Sets the frame counter value, used with OverlayText format code "^c"

`count`
frame counter value

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

S2263_GetFrameCount

```
MID2263_API int __stdcall S2263_SetFrameCount (
      int               *count,
      int               devid);
```

Gets the frame counter value, used with OverlayText format code "^c".

`count`
frame counter value.

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

3364 Specific Functions

S3364_Is3364Board

```
MID2263_API int __stdcall S3364_Is3364Board (
      BOOL        *b3364,
      int         devid);
```

Queries if board 3364 or not.

`b3364`
pointer to boolean determining if 3364 (value = 1) or not (value = 0).

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

S3364_GetFirmwareVersionUSB3

```
MID2263_API int __stdcall S3364_GetFirmwareVersionUSB3 (
      int         *version,
      int         devid);
```

Returns current 3364 USB3 firmware version.

`version`
pointer to integer for version number.

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

S3364_GetUsbSpeed

```
MID2263_API int __stdcall S3364_GetUsbSpeed (
      int         *speed,
      int         devid);
```

Returns speed for USB link for 3364.

`speed`
pointer to integer containing USB speed. *version = 3 corresponds to USB3.0 superspeed.

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

S3364_GetBoardType

```
MID2263_API int __stdcall S3364_GetBoardType (
      int         *type,
      int         devid);
```

Returns 3364 board type. SDI or DVI.

`type`
pointer to integer for board type. A value of 1 corresponds to 3364-SDI, 0 to 3364-DVI.

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

S3364_StartFullSizePreview

```
MID2263_API int __stdcall S3364_StartFullSizePreview (
      int         devid);
```

Starts full size HD uncompressed preview. Requires USB3.0.

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

S3364_StopFullSizePreview

```
MID2263_API int __stdcall S3364_StopFullSizePreview (
      int         devid);
```

Stops full size HD uncompressed preview. Full size preview may also be stopped with S3364_StopStream

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

S3364_FullSizePreviewSample

```
MID2263_API int __stdcall S3364_FullSizePreviewSample (
      unsigned char     *data,
      unsigned int      inlen
      unsigned int      *outlen
      int               timeout
      int               devid);
```

Grabs one frame from the full size preview stream. Full size preview should be running with S3364_StartFullSizePreview first. If preview display not desired while taking samples or snapshots, use the snapshot stream instead via S3364_StartSnapshot and S3364_GetSample. Another alternative for full frame snapshots without preview is to use a callback with S3364_StartCallback, S3364_RegisterCallback.

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

`data`
pointer to supplied buffer

`inlen`
size of data buffer above

`outlen`
number of bytes saved to buffer

`timeout`
timeout to wait if buffer not available. In milliseconds.

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

S3364_StartCallback

```
MID2263_API int __stdcall S3364_StartCallback (
      int         devid);
```

Starts full size callback stream. Use S3364_RegisterCallback to register the callback function. Use S3364_StopStream to stop the callback stream. The

callback stream may not be used at same time as S3364_StartFullSizePreview stream.

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

S3364_RegisterCallback

```
MID2263_API int __stdcall S3364_StartCallback (
      cbfunc_t          callback,
      int               devid);
```

Registers a callback for full size HD capture via callback. Data in the callback will be captured in YUY2 format. Care must be taken to minimize time spent in the callback routine otherwise the buffers used by DirectShow (the Windows capture API) could overflow. Use S3364_StartCallback after registering your callback function.

callback
callback function to use. Callback function should be defined as follows: “int callback_name(BYTE *data, long size, int devid, int strmidx). Note strmidx field in the will always be “3” and may be ignored.

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

S3364_StopStream

```
MID2263_API int __stdcall S3364_StopStream (
      int               devid);
```

Stops a stream (callback, fullsize preview, or snapshot stream) in progress.

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

S3364_StartSnapshot

```
MID2263_API int __stdcall S3364_StartSnapshot (
      int         devid);
```

Starts full size “snapshot” stream. Use S3364_GetSample to get frames. Use S3364_StopStream to end streaming of snapsots

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

S3364_FullSizePreviewSample

```
MID2263_API int __stdcall S3364_FullSizePreviewSample (
      unsigned char     *data,
```

```
      unsigned int      inlen
      unsigned int      *outlen
      int               timeout
      int               devid);
```

Grabs one frame from the full size snapshot stream, started with S3364_StartSnapshot..

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

`data`
pointer to supplied buffer

`inlen`
size of data buffer above

`outlen`
number of bytes saved to buffer

`timeout`
timeout to wait if buffer not available. In milliseconds.

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

S3364_SetUserData

```
MID2263_API int __stdcall S3364_SetUserData (
      char        *data
      int         devid);
```

Saves user data to non-volatile memory on the 3364 board. OEMs or users may save 10 bytes of data to the board.

`data`
pointer to supplied buffer containing data. Must be at least 10 bytes long.

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

S3364_GetUserData

```
MID2263_API int __stdcall S3364_SetUserData (
      char        *data
      int         devid);
```

Retrieve user data from non-volatile memory on the 3364 board.

`data`
pointer to supplied buffer to retrieve data into. Must be at least 10 bytes

long.

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

S2263_GetFullDevicePath

```
MID2263_API int __stdcall S2263_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).

S2263_GetFullDeviceInstance

```
MID2263_API int __stdcall S2263_GetFullDeviceInstance (
      int               nWC
      WCHAR             *dev_instance,
      int               devid);
```

Same as S2263_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).

Revision history

Version	Notes
1.0.0, June 2013	Initial release.
1.0.4, Oct 2014	Updating API
1.0.5, Jan 2015	Overlay functions
1.1.0, Aug 2015	Multiple recording streams, frame skip, fixed QP
1.1.1, May 2016	Output functions
1.1.2, April 2017	3364 support added with extra 3364 specific functions as needed.
1.1.x, June 2018	Added additional OVERLAY_TEXT_DATE formats.