H.264 1080p/1080i/720P Capture Device

Software Manual (Windows)

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

Limited warranty

Sensoray Company, Incorporated (Sensoray) warrants the hardware to be free from defects in material and workmanship and perform to applicable published Sensoray specifications for two years from the date of shipment to purchaser. Sensoray will, at its option, repair or replace equipment that proves to be defective during the warranty period. This warranty includes parts and labor.

The warranty provided herein does not cover equipment subjected to abuse, misuse, accident, alteration, neglect, or unauthorized repair or installation. Sensoray shall have the right of final determination as to the existence and cause of defect.

As for items repaired or replaced under warranty, the warranty shall continue in effect for the remainder of the original warranty period, or for ninety days following date of shipment by Sensoray of the repaired or replaced part, whichever period is longer.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. Sensoray will pay the shipping costs of returning to the owner parts that are covered by warranty. A restocking charge of 25% of the product purchase price, or $105, whichever is less, will be charged for returning a product to stock.

Sensoray believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, Sensoray reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult Sensoray if errors are suspected. In no event shall Sensoray be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, SENSORAY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF SENSORAY SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. SENSORAY WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEROF.

Introduction

The 2226 product is a USB 2.0 audio video H.264 capture device. The 2226 supports many different inputs and video formats including 1080p, 1080i and 720p.

Software Feature Summary

- Outputs MPEG Transport stream with H.264 MPEG encoding.
- Decodes 2226 H264 streams back through hardware decoder.
- Source code for demo provided.
- Full featured demo application including recording and UDP streaming of the stream.
- Free Windows driver(AVStream/DirectShow), 3 demo applications and Software Development Kit (SDK).
- VB.NET demo application, C# (C Sharp) WPF demo application, MFC C++ demo application.

Software

Feature Summary

Sensoray's Model 2226 is shipped with drivers for Microsoft Windows XP, Vista, Windows 7. A full-featured demo application demonstrate capture of the audio/video stream. Control is provided for brightness, bitrate, contrast and other video attributes. Multiple inputs are supported including 1080p(@29.97Hz), 1080p(@30Hz), 1080p(@23.98Hz), 1080p(@24Hz), 1080i(@59.94Hz), 1080i(@60Hz), 1080i(@50 Hz PAL), 720P(@59.94 Hz), 720(@60Hz), 480i(NTSC), 576i(PAL). HDMI is not supported. HD inputs are SDI format.

1080p(@29.97Hz) and 1080p(@30Hz) are not supported for MPEG encode and decode.

The SDK allows maximum flexibility by providing an API for all the 2226's functions. The source code of the demo application is a suggested starting point for custom application development.

Since the 2226 has an AVStream driver, it is DirectShow compliant. Unfortunately, due to the wide number of DirectShow programs, Sensoray cannot guarantee operation with any specific third party program. Please note that CPU usage while previewing(decoding) the stream will be very high due to the high compression of the H.264 stream.

Installation

The software may be distributed on a CD or downloaded from the Sensoray's web site.

Run the setup program from the distribution disk or folder. Software components, including a demo application with the source code, will be installed into the /Program Files/Sensoray/2226 folder.

During the installation the program will pre-install the drivers using Dpinst. Do not click cancel when the driver. Do not plug or unplug the board during the driver installation process.

Redistribution

The SDK CD contains the redistributable targets in the API directory.

The drivers must also be redistributed to end-users and installed for proper function. They are included in the drivers directory after the SDK (setup.exe) is installed.

The DLL uses the Microsoft Visual C++ 2008 runtime. Sensoray's installer will install the C++ runtime files automatically. For customers creating their own redistribution package, the run time package may be downloaded from Microsoft's website.
http://www.microsoft.com/downloads/en/details.aspx?FamilyID=9b2da534-3e03-4391-8a4d-074b9f2bc1bf&displaylang=en

The files PushFileSource2.ax and sraywrite.ax are registered by the installer using regsvr32. If re-distributing using a custom install, these files must be registered on the target computer.

The DLLs alfont.dll, alleg42.dll, mid2226.dll and ovlgen.dll should be included in the same directory as the executable. It is not recommended by Microsoft to install DLLs system wide (in c:\windows\system32).

SDK Reference

Release Notes

V1.0.9 (July, 2013)

- Added 1080p (30/29.9) input with snapshot and overlay on Video outputs.
- Added 1080p (24/23.9) full capability. (snapshot, overlay, and compression/decompression)
- Added 720p (24/23.9) full capability. (snapshot, overlay, and compression/decompression)
- Added TCL interface for prototyping.

V1.0.8 (June, 2013)

- Added and many overlay functions for generating overlays dynamically.
- Increased overlay update speed when dealing with large coverage areas.
- Fixed SDI-embedded audio.
- Added option to display solid Blue or black on input loss.

V1.0.7 (October, 2012)

- Enhanced input loss detection and switchover to blank video output.

V1.0.6 (August, 2012)

- Added input loss detection and switchover to blank video output.

V1.0.5 (June, 2012)

- Added Embedded Audio from SDI input and to SDI output.
- Added hardware level meter for audio.

V1.0.4 (March 8, 2011)

- Added decoding from memory feature (S2226_StartDecodeMem, S2226_SendData).

- Improved demo applications and preview window.
- 64 bit DLL support (in addition to previously released 64 bit driver).
- Decode UDP stream from network added to demo application. (Uses S2226_StartDecodeMem and S2226_SendData internally).

V1.0.3 (November 28, 2011):

- fixes small memory leak on Windows 7

V1.0.2 (November 11, 2011):

- Minor bug fixes

V1.0.1 (August 18, 2011):

- New firmware release. Please download firmware updater from website. Firmware updater is a standalone program. Firmware fixes known issues on V1.0.0 with some SDI cameras. Firmware update is especially important if using the 59.94Hz frame-rate.
- Overlay update speed improved.
- Down-scaling capability on composite outputs. If an SDI HD input is connected, it is possible to downscale in a viewable format on the composite output. Please see S2226_SetOutput.
- Raw preview option on host computer at downscaled resolution. (See S226_SetPreviewType).
- Callback functionality expanded. Callbacks may be used in Preview, Record and Record with Preview streaming as well as standalone callback operation(S2226_StartCallback). See S2226_RegisterCallback. Raw capture callback also possible (S2226_RegisterCallbackRaw).
- Minor bug fixes and new functions.

V.1.0.0 (October 26, 2010):

- Initial release
- V1.0.0 contains decoded preview only. This feature is only available on Windows 7 because XP does not contain the H264 codecs. Please see the section on Preview in the SDK function summary.

A common API flow is described below. Please refer to the Functions Reference and Function summary sections in the manual for more the details on the functions below.

General SDK Usage

1. Enumeration. If using multiple boards, they can be identified with the S2226_Enumerate call. If using a single board, it may be opened without enumeration using the index 0.

2. Initialization. This is performed by a call to `S2226_OpenBoard()` function with the board index parameter. Initial default capture settings are loaded and a handle to the board for other functions is returned.

3. A call to `S2226_OpenBoard()` should be followed by calls to the functions controlling various settings. At a minimum, the correct video input, video system, and clock rate(for HD only) should be set:

    - input source: `S2226_SetInput()`;
    - video system: `S2226_SetVidSys()`, `S2226_SetVidSize()`;
    - advanced video system(for 59.94 HZ NTSC): S2226_SetReducedClock();
    - video parameters (brightness, contrast, saturation, hue): `S2226_SetLevel()`;

4. A call to `S2226_StartRecord()` starts the 2226 and records to file. S2226_StopStream() stops recording from the 2226. The demo application also shows how to capture the data for other purposes using the callback feature. Do not unplug the input signal before `S2226_StopStream()` is called.

5. During the recording the following function could optionally be used to obtain some useful information.

    - `S2226_GetStatus()` – current status, current recorded file size and path;

6. `S2226_CloseBoard()` must be called before application terminates to clean up properly and close the board handle.

Demo applications

The SDK includes a demo application provided with the source code to illustrate the use of SDK's functions.

Function Summary

Initialization

Initialization is done by opening the board. If unsure what board to open (in the case of multiple boards), use the enumerate function S2226_Enumerate to find detect the number of boards and associate each board index with a serial number. If using a single board, it is possible just to open it with S2226_OpenBoard. Before any other function in the SDK can be used (except S2226_Enumerate), the board must be opened with S2226_OpenBoard. This function gives a handle to the board, which is used for the other functions. Only one instance of the board should be opened at one time.

After the board is successfully opened, it is necessary to configure it. Generally, it is not necessary to configure the audio settings. The default audio settings are usually adequate. If required, the audio may be adjusted more easily by temporarily changing the audio route to bypassed using S2226_SetAudioRoute. This mode passes through the audio from the input to the output audio connector.

The input, clock(if HD), and video system must be correctly configured before any encoding function such as S2226_StartRecord is started. Please see the function reference for a detailed description of these functions.

Encoding

- S2226_SetBitrate

Video

- S2226_SetVidSys, S2226_GetVidSys
- S2226_SetInput, S2226_GetInput
- S2226_CheckInput
- S2226_SetReducedClock (HD only), S2226_GetReducedClock (HD only)
- S2226_SetLevel, S2226_GetLevel

Audio

- S2226_SetAudioAgc, S2226_GetAudioAgc
- S2226_SetAudioGain, S2226_GetAudioGain
- S2226_SetAudioBalanced, S2226_GetAudioBalanced
- S2226_SetAudioOutputVol, S2226_GetAudioOutputVol
- S2226_SetAudioOutputMono, S2226_GetAudioOutputMono
- S2226_SetAudioOutputStereo, S2226_GetAudioOutputStereo
- S2226_SetAudioOutputHp, S2226_GetAudioOutputHp
- S2226_SetAudioRoute, S2226_GetAudioRoute
- S2226_SetAudioMuxMpegIn
- S2226_SetAudioMuxLineOut
- S2226_SetAudioMuxSdiOut

Audio Meter

- S2226_SetAudioMtrHoldTime
- S2226_GetAudioMtrHoldTime
- S2226_SetAudioMtrHoldRelease
- S2226_GetAudioMtrHoldRelease
- S2226_SetAudioMtrTest
- S2226_GetAudioMtrTest
- S2226_SetAudioMtrChnSel
- S2226_GetAudioMtrChnSel
- S2226_GetAudioMtrLevel
- S2226_GetAudioMtr_dB
- S2226_GetAudioMtrHold

Recording

There are two functions to record the stream to file. One version with preview and one without. Please note that Windows7 does not allow recording to the root drive or in the "Program Files" directory unless the application is run as administrator.

The functions for recording are shown below. If using S2226_StartPreviewAndRecord or S2226_StartPreviewAndRecordW, please read the section on preview. Before calling any of these functions, a valid input must be connected to the 2226 and the configuration settings properly configured. The 2226 does not support switching inputs while streaming. Do not unplug the input after recording has started.

- S2226_StartRecord, S2226_StartRecordW(unicode)
- S2226_StartRecordAndPreview, S2226_StartRecordAndPreviewW
- S2226_StopStream

Stream Capture

Stream capture is the process of capturing the encoded H.264 stream to memory. This is done by a callback mechanism. The callback must not block and should return in a timely manner. If the callback does no return fast enough, data will be lost and the stream could be corrupted. If a lot of processing must be done on the data during the callback, it is recommended to use standard software Engineering

techniques to work with the data. This could involve, for example, saving the data from the callback to a queue or FIFO to work on in another thread.

An example of stream capture using callbacks is shown in the demo application. It captures the stream and sends it out on a UDP socket. Because the 2226 captures encoded transport stream, no processing or headers need to be added. The resulting stream can be viewed on the VideoLan media player by opening a network stream with arguments such as udp://@:1234 (for UDP destination port 1234).

The functions associated with stream capture through callbacks are shown below. The 2226 does not support switching inputs while streaming. Do not unplug the input after streaming has started.

In version 1.0.1, callbacks can be registered for any streaming function such as recording. For instance, you can set a callback with S2226_RegisterCallback and then call S2226_StartRecord. This will record the stream to file and give a callback to the data. After S2226_StopStream is callback, the callback is unregistered.

Also in version 1.0.1 is a callback to the raw preview stream. This is available by using the function S2226_RegisterCallbackRaw.

- `S2226_RegisterCallback`
- `S2226_RegisterCallbackRaw`
- `S2226_StartCallback (callback stream only)`
- S2226_StopStream

Preview

Preview is the display of the stream on a host computer or PC. The 2226 also has output channels to display the connected input on an external monitor, but these are considered a different feature.

Version 1.0.1 has a raw preview feature on the host computer (at reduced resolution to meet USB bandwidth requirements). Raw preview is at 320x240 or 640x240 resolution and can be set using S2226_SetPreviewType. Raw preview is down-scaled from the selected input in the hardware.

Due to codec issues, decoded preview is only available on Windows 7. Additionally, the following caveats apply to Preview with Recording (S2226_StartPreviewAndRecord). DirectShow has a known issue where the video preview is restarted when the Window is moved from one monitor to another. This is normally not a problem, but if the underlying stream graph is also recording the stream, data will be lost from the recorded file.

There is a workaround to the above problem associated with S2226_StartPreviewAndRecord. The solution is to prevent the user from moving the application Window from one monitor to another while S2226_StartPreviewAndRecord is running. An example is shown in the VB.NET demo. Please note if the stream is stopped, moving the application from one monitor to another is perfectly acceptable.

Additionally, Sensoray has decided that loss of recording data is much more serious than loss of preview on the PC. Therefor, Sensoray has overridden the behavior of DirectShow to prevent this from happen.

Unfortunately, if the user does manage to move the video window from one display to another, the preview may freeze. When this happens, the recording will continue (file size will still increase) but the user may be alarmed by the loss of preview. This at least gives the option of stopping the stream and restarting with a new file without losing recorded data. It is best to prevent this in the first place by using the workaround to prevent the video preview from changing monitors.

H264 is a highly compressed format. When using a slower PC, high CPU usage and preview stuttering while previewing the stream may occur. In this case, it is best to use an external monitor and the 2226 hardware outputs to preview the stream.

Raw preview is much less CPU intensive. This eliminates the multi-monitor issue above at the tradeoff of reduced resolution.

Callbacks are also allowed during preview.

The following functions described later in the Functions Reference are associated with Preview. The 2226 does not support switching inputs while streaming. Do not unplug the input after streaming has started.

- `S2226_StartPreview`
- `S2226_StartPreviewAndRecord, S2226_StartPreviewAndRecordW`
- `S2226_StopStream`

Decoding

The 2226 has the capability to decode streams recorded by the same hardware. It is a closed decoder in that other H.264 streams not recorded by the 2226 codec may not be decoded successfully. The decoder must know the format (video size and clock) of the stream. It does not have the capability to auto-detect the stream format parameters. Please see the function reference for the functions below:

- S2226_StartDecode, S2226_StartDecodeW, S2226_StartDecodeMem, S2226_SendData
- S2226_StopStream

Overlay

The 2226 has the capability to add text captions and bitmap images on to the video stream. There are a total of 8 hardware windows to work with. Please note that multi-line captions may take up multiple windows. The demo application currently only demonstrates one overlay region.

All overlay functions (except S2226_CopyBmpToOverlayZero) do not update the screen until S2226_UpdateOverlay is called. If using multiple overlays, set them up first and then call S2226_UpdateOverlay for maximum efficiency. S2226_CopyBmpToOverlayZero can be called to update sub-regions of overlay window-0 immediately. When window-0 is set to the same size as the input resolution, this effectively enables random rectangular updates across the entire screen without having to download the entire overlay, as is done with the other functions when updated with S2226_UpdateOverlay.

- S2226_OverlayText, S2226_OverlayTextIdx
- S2226_OverlayImage, S2226_OverlayImageIdx
- S2226_GetOverlayIdx
- S2226_OverlayBackgroundColor
- S2226_MoveOverlay
- S2226_OverlayDel, S2226_OverlayDelXY
- S2226_ClearOverlay
- S2226_UpdateOverlay
- S2226_ClearOverlayRegion
- S2226_SetOverlayRegion
- S2226_OverlayImageRaw
- S2226_CopyBmpToOverlayZero

Snapshots

Snapshots are uncompressed grabs from the 2226 in near real-time. They are captured directly in hardware at the same video resolution as the input. They are not decoded from the encoded stream so there will be no loss of quality due to compression. Due to USB bandwidth limitations, the snapshot feature is only intended for intermittent grabs of the stream. The user must not attempt to capture at anywhere near full frame rate using snapshots.

Snapshots are captured field by field. If using an interlaced source, the merge type should be specified. The merge type determines how to put the fields back together. Please see the function reference for more details.

- S2226_SetMergeMethod
- S2226_SnapshotToFile
- S2226_SnapshotToFileW
- S2226_SnapshotToMem
- S2226_SnapshotRaw

Notifications

The 2226 is a USB device. It should not be unplugged during streaming or when the application is running. However, sometimes this may happen. The functions S2226_SetNotify and S2226_TestDeviceRemoval are used to see if the device was unplugged. The demo application shows how these functions are used. The demo application will be closed if the 2226 is harshly unplugged while the demo is running.

- S2226_SetNotify
- S2226_TestDeviceRemoval

Cleanup/Shutdown

After all work is done with the 2226, the SDK should be closed for that board. An example is when the demo application closes. The following function closes the SDK for an open board.

- S2226_CloseBoard

Functions Reference

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

#define MID2226_API extern "C" __declspec(dllimport)

```
MID2226_API HANDLE S2226_OpenBoard(int devid);
```

Must be called before any other API functions are called to open the SDK.

Parameters

devid

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

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_CloseBoard(HANDLE hdev);
```

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

Parameters

hdev

handle to device.

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_Enumerate(
   int *count,
   DEVINFO2226 *devices
);
```

Enumerates all plugged in 2226s in the system. If *pCount equal to 0, the number of attached devices is set in *pCount. If *pCount != 0 and pDevices != NULL, then pDevices points to a list of at least *pCount DEVINFO2226 structures which get filled in with board number and serial number information. Please see demo application for an example.

Parameters

count

Returns devices found in system (if called with *pCount=0) or size of pDevices to fill with board information.

devices

array of at least *pCount devices. If querying the number of devices (*pCount = 0), pDevices may be NULL.

Returns

0 if success, negative if error.

```
MID2226_API int S2226_SetInput(
   MID2226_SOURCE input,
   BOOL bUpdateNow
   HANDLE hdev
```

```
);
```

Selects current input. If bUpdateNow is set to FALSE, the input will not be updated until streaming is started. If changing from an HD input to anything else(including a different HD input), some settings may be reset to default. Please note that input changes are not allowed during streaming. Always stop the stream before changing inputs.

Parameters

input

enumerated input MID2226_INPUT (see mid2226types.h). The allowed values are:

- MID2226_INPUT_COMPOSITE_0 (Main composite input for SD)
- MID2226_INPUT_SVIDEO_0 (Svideo input for SD)
- MID2226_INPUT_COMPOSITE_1 (Alternate composite input via header)
- MID2226_INPUT_SVIDEO_1 (Alternate svideo input)
- MID2226_INPUT_SD_COLORBARS (Not a physical input. Test input for SD)
- MID2226_INPUT_720P_COLORBARS(Not a physical input. Test input for 720p HD)
- MID2226_INPUT_1080I_COLORBARS(Not a physical input. Test input for 1080i HD)
- MID2226_INPUT_SDI_SD (SDI input with standard definition source)
- MID2226_INPUT_SDI_720P (SDI input with 720p HD source. 59.94Hz, 60Hz only)
- MID2226_INPUT_SDI_1080I (SDI input with 1080I HD source: 50Hz, 59.94Hz, 60Hz)
- MID2226_INPUT_1080P30_COLORBARS (for snapshot and overlay only)
- MID2226_INPUT_1080P24_COLORBARS
- MID2226_INPUT_720P24_COLORBARS
- MID2226_INPUT_SDI_1080P30 (for snapshot and overlay only)
- MID2226_INPUT_SDI_1080P24
- MID2226_INPUT_SDI_720P24

bUpdateNow

Set to TRUE to change input immediately (if stream stopped). Otherwise input changed at next stream start. If bUpdateNow is set to TRUE and new input has a different video system, call S2226_SetVidSys first or the new video system will not be updated until the stream is started.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetInput(
   MID2226_SOURCE *pSource,
   HANDLE hdev
);
```

Retrieves current input settings.

Parameters

pSource

pointer to the value to receive the current input.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_CheckInput(
   MID2226_SOURCE *ilst,
   HANDLE hdev
);
```

Retrieves current input settings.

Parameters

ilst

pointer to the value of current input status. Will be set to 1 if the current input is lost.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetBitrate(
   unsigned long bitrate,
   HANDLE hdev
);
```

Retrieves current input settings.

Parameters

bitrate

bitrate of encoded stream in kbps. (1000-17000kbps).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetVidSys(
   MID2226_VIDSYS vidsys,
   HANDLE hdev
);
```

Sets the input video system (NTSC, PAL).

Parameters

vidsys

video system enumerated type (see mid2226types.h).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetVidSys(
   MID2226_VIDSYS *pvidsys,
   HANDLE hdev
);
```

Gets the input video system (NTSC, PAL).

Parameters

pvidsys

pointer to video system enumerated type (see mid2226types.h).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetReducedClock(
   BOOL bReducedClock,
   HANDLE hdev
);
```

For HD inputs only (720p, 1080i) with NTSC video system. Some HD NTSC sources are at 59.94Hz and others at 60Hz. Allows changing between the two types. Note, this is a required setting for HD inputs with the NTSC video system.

Parameters

bReducedClock

if 1, video clock is at 59.94Hz, otherwise clock is 60Hz.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetReducedClock(
   BOOL *bReducedClock,
   HANDLE hdev
);
```

Retrieves reduced clock value.

Parameters

bReducedClock

if 1, video clock is at 59.94Hz, otherwise clock is 60Hz.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetStatus(
   MID2226STATUS *pStatus,
   HANDLE hdev
);
```

Retrieves current status information (see MID2226func.h for `MID2226STATUS` type definition). Multibyte (ASCII) filenames.

Parameters

pStatus

pointer to status variable.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetStatusW(
   MID2226STATUS_W *pStatus,
   HANDLE hdev
);
```

Same as S2226_GetStatus, but uses Unicode filenames.

```
MID2226_API int S2226_StartRecord(
   char *fileName,
   HANDLE hdev
);
```

Starts recording to a file.

Parameters

fileName

full path to the target file, no extension.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartRecordW(
   wchar_t *fileName,
   HANDLE hdev
);
```

Same as S2226_StartRecord, but uses widechar or Unicode filenames.

```
MID2226_API int S2226_StopStream(
   HANDLE hdev
);
```

Stops streaming (recording, playing, previewing). Any registered callback is cleared.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetLevel(
   int param,
   int value,
   HANDLE hdev
);
```

Sets brightness, contrast, saturation and hue of the captured video.

Parameters

param

defines the parameter to set (`MID2226_LEVEL_CONTRAST, MID2226_LEVEL_BRIGHTNESS, MID2226_LEVEL_SATURATION, MID2226_LEVEL_HUE`). See see mid2226types.h for definitions.

value

defines the value of selected parameter

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetLevel(
   int param,
   int value,
```

```
   HANDLE hdev
);
```

Retrieves video levels.

Parameters

param

defines the parameter to get (`MID2226_LEVEL_CONTRAST`, `MID2226_LEVEL_BRIGHTNESS`, `MID2226_LEVEL_SATURATION`, `MID2226_LEVEL_HUE`). See see mid2226types.h for definitions.

value

pointer to returned value of selected parameter

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioGain(
   int gainL,
   int gainR,
   HANDLE hdev
);
```

Sets gain settings for internal PGA audio amp.

Parameters

gainL

gain for left channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input

gainR

gain for right channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioGain(
   int *gainL,
   int *gainR,
   HANDLE hdev
);
```

Retrieves gain settings for internal PGA audio amp.

Parameters

gainL

gain for left channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input

gainR

gain for right channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioAgc(
   int bOnL,
   int bOnR,
   int gainL,
   int gainR,
   HANDLE hdev
);
```

Sets gain settings for audio automatic gain control.

Parameters

bOnL

toggles AGC gain for left channel. 0(off) recommended for line input.

bOnR

toggles AGC gain for right channel. 0(off) recommended for line input.

gainL

AGC gain for left channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input.

gainR

AGC gain for right channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioAgc(
   int *bOnL,
   int *bOnR,
   int *gainL,
   int *gainR,
   HANDLE hdev
);
```

Retrieves gain settings for audio automatic gain control.

Parameters

bOnL

toggles AGC gain for left channel. 0(off) recommended for line input.

bOnR

toggles AGC gain for right channel. 0(off) recommended for line input.

gainL

AGC gain for left channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input.

gainR

AGC gain for right channel(decibels times 2). 0-118 (0-59dB) 0dB recommended for line input.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioBalanced(
   BOOL bBalanced,
   HANDLE hdev
);
```

Sets whether audio input is balanced(differenced) or not (default).

Parameters

bBalanced

1 if input is balanced, 0 otherwise.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioBalanced(
   BOOL *bBalanced,
   HANDLE hdev
);
```

Retrieves audio balanced setting.

Parameters

bBalanced

1 if input is balanced, 0 otherwise.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioOutputVol(
```

```
   int val,
   HANDLE hdev
);
```

Used for S2226_StartDecode only. Set the audio DAC volume.

Parameters

val

0-127 (steps of -0.5 dB, 0=0Db=maximum volume, 127=-63.5dB).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioOutputVol(
   int *val,
   HANDLE hdev
);
```

Retrieves audio output volume setting.

Parameters

val

0-127 (steps of -0.5 dB, 0=0Db=maximum volume, 127=-63.5dB).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioOutputMono(
   int extra_gain,
   HANDLE hdev
);
```

Sets extra gain on the audio mono channel.

Parameters

extra_gain

0-9 (in steps of 1 dB). The default and recommended setting is 0.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioOutputMono(
   int *extra_gain,
   HANDLE hdev
);
```

Retrieve extra gain setting for audio mono channel.

Parameters

extra_gain

0-9 (in steps of 1 dB). The default and recommended setting is 0.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioOutputHp(
   int extra_gain,
   HANDLE hdev
);
```

Sets extra gain on the audio high power(HP) channel.

Parameters

extra_gain

0-9 (in steps of 1 dB). The default and recommended setting is 0.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioOutputHp(
   int *extra_gain,
   HANDLE hdev
);
```

Retrieve extra gain setting for audio high power(HP) channel.

Parameters

extra_gain

0-9 (in steps of 1 dB). The default and recommended setting is 0.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioOutputStereo(
   int extra_gain,
   HANDLE hdev
);
```

Sets extra gain on the audio high stereo channel.

Parameters

extra_gain

0-9 (in steps of 1 dB). The default and recommended setting is 0.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioOutputStereo(
   int *extra_gain,
   HANDLE hdev
);
```

Retrieve extra gain setting for audio stereo channel.

Parameters

extra_gain

0-9 (in steps of 1 dB). The default and recommended setting is 0.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioRoute(
   int route,
   HANDLE hdev
);
```

Sets audio route. For debug or setup only. Do not leave audio route on bypassed while recording. If route is bypassed, then the audio from the input is directly connected to the output.

Parameters

route

0 – normal audio input, 1 – unused, 2 - bypassed input, 3 - unused

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioRoute(
   int *route,
   HANDLE hdev
);
```

Gets the current audio routing. For debug or setup only.

Parameters

route

0 – normal audio input, 1 – unused, 2 - bypassed input, 3 - unused

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioMuxLineOut(
   MID2226_MUX_LINE_OUT_TYPE source,
   HANDLE hdev
);
```

Select Source feeding the audio Line out.

Parameters

source

Source can be one of:

AMUX_LINE_OUT_LINE_IN - Line-Out gets Line-In

AMUX_LINE_OUT_MPEG_OUT - Line-Out gets MPEG-Out

AMUX_LINE_OUT_SDI_IN - Line-Out gets SDI-In embedded audio

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioMuxMpegIn(
   MID2226_MUX_MPEG_IN_TYPE source,
   HANDLE hdev
);
```

Select Source feeding the MPEG audio to be encoded.

Parameters

source

Source can be one of:

AMUX_MPEG_IN_LINE_IN - MPEG-In gets Line-In

AMUX_MPEG_IN_TONE - MPEG-In gets Tone Test

AMUX_MPEG_IN_SDI_IN - MPEG-In gets SDI-In embedded audio

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioMuxSdiOut(
   MID2226_MUX_SDI_OUT_TYPE source,
   HANDLE hdev
);
```

Select Source feeding the SDI-Out embedded audio encoder.

Parameters

source

Source can be one of:

AMUX_SDI_OUT_LINE_IN - SDI-Out gets Line-In

AMUX_SDI_OUT_MPEG_OUT - SDI-Out gets MPEG-Out

AMUX_SDI_OUT_TONE - SDI-Out gets Tone Test

AMUX_SDI_OUT_SDI_IN - SDI-Out gets SDI-In embedded audio

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioMtrChnSel(
   int source,
   HANDLE hdev
);
```

Set Audio Meter Channel Select. Determines which Audio channel to send to the meter/monitor hardware.

Parameters

source

Source can be one of:

0 = Line-in audio channel

1 = Mpeg-out audio channel

2 = SDI-in audio channel

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioMtrChnSel(
   int *source,
   HANDLE hdev
);
```

Read the current source setting feeding the Audio Meter.

Parameters

source

pointer to the value to receive the current Audio Meter input setting.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioMtrHoldRelease(
   int holdrel,
   HANDLE hdev
);
```

Set Audio Meter's Hold/Release setting.

Parameters

holdrel

Can be one of:

1 = Force release of 'Held' output

0 = Allow holding of highest db value per the set hold time

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioMtrHoldRelease(
   int *holdrel,
   HANDLE hdev
);
```

Read the current Hold/Release setting.

Parameters

holdrel

pointer to the value to receive the current Hold/Release setting.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioMtrHoldTime(
   int val,
   HANDLE hdev
);
```

Set Audio Meter Hold time. This is the time the current highest value is saved before being replaced with a lower/next highest value.

Parameters

val

Can be one of:

0 = no hold (Update every 1 ms)

1 = Hold hold highest value 0.5 sec

2 = Hold hold highest value 1.0 sec

3 = Hold hold highest value 1.5 sec

4 = Hold hold highest value 2.0 sec

5 = Hold hold highest value 2.5 sec

6 = Hold hold highest value 3.0 sec

7 = hold forever

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioMtrHoldTime(
   int *val,
   HANDLE hdev
);
```

Read the current hold time setting.

Parameters

val

pointer to the value to receive the current hold time setting.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetAudioMtrTest(
   int val,
   HANDLE hdev
);
```

Set Audio Meter test type.

Parameters

val

Can be one of:

0 = No Test.

1 = Force to zero

2 = Force to clip

3 = Force to -6dB

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioMtrTest(
   int *val,
   HANDLE hdev
);
```

Read the current Audio Meter test type.

Parameters

val

pointer to the value to receive the current test type setting.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioMtrHold(
   int *hld_l,
   int *hld_r,
   int *clip_l,
   int *clip_r,
   HANDLE hdev
);
```

Read the current Audio Meter maximum values being held. Left and right hold values are 11-bits unsigned binary, where 0 is the max volume. Each step is worth -0.1 db and there are 2048 steps

The clipping values indicate that an audio value was clipped and that the maximum positive or negative value was detected. (0x7FFFFF or 0x800000 detected after sign extension.)

Parameters

hld_l

pointer to the value to receive the left channels current held value.

hld_r

pointer to the value to receive the right channels current held value.

clip_l

pointer to the value that will be set to 1 when clipping is detected on the left channel.

clip_r

pointer to the value that will be set to 1 when clipping is detected on the right channel.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioMtrLevel(
   int *audl,
   int *audr,
   HANDLE hdev
);
```

Get the audio meter's peak amplitude that is decayed over time. Left and right values are 23-bits, unsigned binary , where 0 indicates the min volume.

Parameters

audl

pointer to the value to receive the left channels current peak amplitude that is decayed over time.

audr

pointer to the value to receive the right channels current peak amplitude that is decayed over time.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetAudioMtr_dB(
   int *db_l,
   int *db_r,
   HANDLE hdev
);
```

Get the audio meter's peak amplitude, in decibels, that is decayed over time. Left and right values are 11-bits, unsigned binary, where 0 indicates the max volume. There are 2048 steps of -0.1 db each.

Parameters

db_l

pointer to the value to receive the left channels current peak amplitude that is decayed over time in decibels.

db_r

pointer to the value to receive the right channels current peak amplitude that is decayed over time in decibels.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartCallback(
   HANDLE hdev
);
```

Starts streaming with H264 data sent to the callback function registered with S2226_RegisterCallback function. After S2226_StopStream, callback in reset. Callbacks should be short and return quickly.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_RegisterCallback(
   cbfunc_t callback_function,
   HANDLE hdev
);
```

Registers a callback function. Call before S2226_StartPreview, S2226_StartRecord, S2226_StartPreviewAndRecord or S2226_StartCallback to receive the data as it is captured. Data available in the callback is the compressed MPEG H.264 transport stream.

To turn off callbacks, call with an argument of NULL for the callback function.

The callback is for the MPEG stream only.

MPEG data is received from the board in a bursty manner. As such, the callback may be called at any time during streaming. Callbacks must be short and return quickly. If not, buffer the data and process it in another thread. Do not call any stream control functions in the callback

Parameters

callback_function

callback function. See header file for definition of cbfunc_t function.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_RegisterCallbackRaw(
   cbfunc_t callback_function,
   HANDLE hdev
);
```

Registers a callback function. Call before S2226_StartPreview, S2226_StartRecord, S2226_StartPreviewAndRecord or S2226_StartCallback to receive the data as it is captured. Data available in the callback is the raw preview downscaled stream (320x240). The callback is unregistered after S2226_StopStream or S2226_StopRawPreview. Each callback will occur on capture of a single complete frame (but not necessarily synchronized to the MPEG stream).

Parameters

callback_function

callback function. See header file for definition of cbfunc_t function.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartRawPreviewCallback(
   HANDLE hdev
);
```

This is similar to S2226_StartRawPreview, but it does not render the image (display it) on the screen. It is designed to be used in conjunction with S2226_RegisterCallbackRaw in order to grab raw frames faster (by streaming) instead of using the other S2226_ snapshot functions. Please see the section S2226 Snapshot demo later in this manual.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartDecode(
   char *fileName,
   MID2226_DECODE_TYPE decode_type,
   HANDLE hdev
);
```

Starts decoding a file stream on the 2226 hardware. Output will not be on host or PC. Output will be from physical 2226 hardware connectors. If the stream (decode_type) is HD (720p or 1080i), there will be no valid output from the composite outputs on the 2226. The 2226 is not full duplex so S2226_StartDecode may not be called while recording or performing callback streaming.

Parameters

fileName

full path to the target file, no extension.

decode_type

decode type. See mid2226types.h.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartDecodeW(
   wchar_t *fileName,
   HANDLE hdev
);
```

Same as S2226_StartDecode, but uses widechar or Unicode filenames.

```
MID2226_API int S2226_StartDecodeMem(
   MID2226_DECODE_TYPE decode_type,
   HANDLE hdev
);
```

Starts decoding a stream from memory. See demo application for an example usage of this function. Use S2226_SendData to send data to the 2226 to decode.

Parameters

decode_type

decode type. See mid2226types.h.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SendData(
   unsigned char *data,
   int len,
   HANDLE hdev
);
```

Sends data to the board for decoding.

Parameters

data

pointer to data

len

length of data (must be greater or equal to 8192)

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_PauseDecode(
   HANDLE hdev
);
```

If a decode is in progress, this function will pause the output.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_NormalDecode(
   HANDLE hdev
);
```

If a decode is in progress and was paused or running in the slow mode, this function will resume normal output operation at full speed.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SlowDecode(
   int hold,
   HANDLE hdev
);
```

If a decode is in progress, this function will start slow motion decode. The hold time controls how long each successive frame is displayed.

Parameters

hold

hold time is the number of ½ frames to hold each frame on decode. Use 4(half speed) to 255 (slowest).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_TestDecodeDone(
   HANDLE hdev
);
```

If message received, tests if decode was finished in the driver. This is NOT, however, when decode stops on the codec chip. It is only when the driver is finished with the data. A future update will correct this limitation.

Because the data is compressed and there are buffers on the board, TestDecodeDone may be early by a substantial amount.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetPreviewType(
   MID2226_PREVIEWTYPE type,
   HANDLE hdev
);
```

Sets the preview type (on the host computer, NOT the 2226 physical outputs). Raw preview is downscaled to 320x240 or 640x480 resolution to meet USB2.0 bandwith requirements (combined with the MPEG stream).

Parameters

type

Preview type (raw or decoded MPEG stream)

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartPreview(
   HWND hwnd,
   HANDLE hdev
```

```
);
```

Starts video preview achieved by decompressing the stream. Available on Windows 7 only. Please note that H264 requires significant CPU resources to decode. Please see preview in the function summary for more information about preview.

Parameters

hwnd

handle to display video in (use NULL for pop-up window).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartPreviewAndRecord(
   HWND hwnd,
   char *fileName,
   HANDLE hdev
);
```

Starts recording to a file with video preview achieved by decompressing the stream. Available on Windows 7 only. Please note that H264 requires significant CPU resources to decode. Please see preview in the function summary for more information about preview.

Parameters

hwnd

handle to display video in (use NULL for pop-up window).

fileName

full path to the target file, no extension.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartPreviewAndRecordW(
   HANDLE hwnd,
```

```
   wchar_t *fileName,
   HANDLE hdev
);
```

Same as S2226_StartPreviewAndRecord, but uses widechar or Unicode filenames.

```
MID2226_API int S2226_SetRawPreviewSize(
   int size
   HWND hwnd,
   char *fileName,
   HANDLE hdev
);
```

Sets the size of the uncompressed (Raw) preview window. This must be called before the preview is started. Any scaling above 640x480 will be at reduced frame rate due to USB2.0 bandwidth limitations. 1280x720 is not available for 1080 inputs. 1920x1080 is not available for 720P inputs. The larger scalings require the latest firmware from Sensoray. The firmware update may be updated from the Sensoray website free of charge.

Parameters

size

Size of the uncompressed (Raw) preview window. Valid values are MID2226_RAWPREVIEW_320_240, MID2226_RAWPREVIEW_640_480,

MID2226_RAWPREVIEW_1280_720, MID2226_RAWPREVIEW_1920_1080

hwnd

handle to display video in (use NULL for pop-up window).

fileName

full path to the target file, no extension.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StartRawPreview(
   HWND hwnd,
   char *fileName,
   HANDLE hdev
);
```

Start raw preview allows viewing the raw preview while streaming. It can be used with S2226_StartRecord or S2226_StartCallback. It should not be used with the S2226_StartPreview functions.

Use S2226_StopRawPreview to stop the independent preview stream. All streams must be stopped before the input can be changed.

If the input was changed and S2226_StartRawPreview is still running, then the other S2226_Start functions will fail until S2226_StopRawPreview is called.

Parameters

hwnd

handle to display video in (use NULL for pop-up window).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_StopRawPreview(
   HANDLE hdev
);
```

Stop a running raw preview window.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_Repaint(
   HDC hdc,
   HANDLE hdev
);
```

Only used for preview and when rendering to a non-NULL hwnd. Call this function in your OnPaint routine. See demo application for example usage.

Parameters

hdc

handle to device context.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_DisplayChange(
   HDC hdc,
   HANDLE hdev
);
```

Call this function in your WM_DISPLAYCHANGE message handler for the video window. See demo application for example usage.

Parameters

hdc

handle to device context.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetNotify(
   HWND hNotifyWnd,
   UINT mNotifyMessage
   HANDLE hdev
);
```

Use to set up a callback to the HWND when device message occurs. Use in conjunction with S2226_TestDeviceRemoval to see when device removed (see demo application for implementation details).

Parameters

hNotifyWnd

Window to notify on device event (removal)

mNotifyMessage

Window message to use for notification.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_TestDeviceRemoval(
   HANDLE hdev
);
```

After receiving a message (set up by S2226_SetNotify), call S2226_TestDeviceRemoval to see if device was removed (unplug).

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

```
MID2226_API int S2226_OverlayText(
   int xpos,
   int ypos,
   overlay_text_t *ovltext,
   int regionmask,
   HANDLE hdev
);
```

Adds overlay text. If overlay text already exists at that x,y position, deletes windows before adding. Overlay active on regions defined by regionmask. If text contains embedded newline characters (\n = 10 dec), then each line of text will be created on in a new window, AtIndex sub-window position, directly below the preceding line. Each sub-window position only consumes enough overlay memory needed to hold the individual line. If text contains embedded character 30 dec (entered programmatically or by holding down Alt- and typing "030" on the numeric keypad), then each line following a char(30) will be on a new line. Multi-line text created this way will be created as one large graphic at one "index" location.

Parameters

xpos

start x position.

ypos

start y position.

ovltext

pointer to overlay text.

regionmask

MID2226_REGION_MONITOR, MID2226_REGION_MPEG, MID2226_REGION_STILL.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, -1 if too many regions, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_OverlayTextIdx(
   int AtIndex,
   int xpos,
   int ypos,
   overlay_text_t *ovltext,
   int regionmask,
   HANDLE hdev
);
```

Adds overlay text. If overlay already exists at index WinIndex, deletes window before adding. See S2226_OverlayText for more details about the overlay and multi-line support.

Parameters

AtIndex

(0-7) sub window position to update/add text x position.

xpos

start x position.

ypos

start y position.

ovltext

pointer to overlay text.

regionmask

MID2226_REGION_MONITOR, MID2226_REGION_MPEG, MID2226_REGION_STILL.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_OverlayImage(
   int xpos,
   int ypos,
   char *imageFile,
   int regionmask,
   HANDLE hdev
);
```

Adds overlay image. If overlay already exists at that x, y position, deletes window before adding.

Parameters

xpos

start x position.

ypos

start y position.

imageFile

full path to image file. Must be a 24bit BMP or PCX bitmap image only. See demo application and Logo.bmp.

regionmask

MID2226_REGION_MONITOR, MID2226_REGION_MPEG, MID2226_REGION_STILL.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, -1 if too many regions, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_OverlayImageIdx(
   int AtIndex,
   int xpos,
   int ypos,
   char *imageFile,
   int regionmask,
   HANDLE hdev
```

```
);
```

Adds overlay image. If overlay already exists at that x, y position, deletes window before adding.

Parameters

AtIndex

(0-7) sub-window position to update/add text.

xpos

start x position.

ypos

start y position.

imageFile

full path to image file. Must be a 24bit BMP or PCX bitmap image only. See demo application and Logo.bmp.

regionmask

MID2226_REGION_MONITOR, MID2226_REGION_MPEG, MID2226_REGION_STILL.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_OverlayImageRaw(
   BYTE *image,
   int xPos,
   int yPos,
   int xSize,
   int ySize,
   int AtIndex,
   int regionmask,
   HANDLE handle
);
```

Adds/Update overlay image. If overlay already exists at index WinIndex, deletes window before adding.

Image

Pointer to memory containing 24-bit RGB data Row 0 first

xPos

Start X position

yPos

Start y position

xSize

Size of horizontal line in pixels (Each pixel is 3 bytes: LSB=Red,Green,MSB=Blue)

ySize

Number of Lines

AtIndex

Sub-window position to update/add image

regionmask

Output region mask

board

ID if multiple boards installed.

Returns

0 on success, -1 on too many regions, other negative value on failure

```
MID2226_API int S2226_GetOverlayIdx(
   int WinIndex,
   int *type,
   int *region,
   int *group,
   int *xpos,
   int *ypos,
   char **value,
   HANDLE hdev
);
```

Get basic parameters of an overlay index.

Parameters

WinIndex

(0-7) sub-window position to query.

type

1=image, 0=text

region

MID2226_REGION_MONITOR, MID2226_REGION_MPEG, MID2226_REGION_STILL.

group

text with common group number is kept together.

xpos

start x position.

ypos

start y position.

value

pointer to text or image file path.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_CopyBmpToOverlayZero(
   char *ImageFile,
  char *Image,
  int xPos,
  int yPos,
  int xSize,
  int ySize,
  int Backgnd_ForegndN,
  HANDLE handle
);
```

Copy given image to region within sub-window position zero.

- Assumes sub-window position zero is loaded into current memory
- Assumes sub-window position zero has a width that is divisible by 8
- Assumes all other overlays (1-7) are contained within the boundaries of overlay zero

NOTE: All conditions are met when sub-window position 0 is set to match the input video resolution, thereby creating a full screen overlay that can be quickly updated immediately with this call. All other overlay functions use S2226_UpdateOverlay() to calculate and download ALL sub-windows each time it is called.

ImageFile

Full name/path to image file. 24-bit bmp and pcx files are supported. If NULL, use Raw 24-bit bitmap pointer Image

ImageRGBptr

Pointer to memory containing 24-bit RGB data Row 0 first

xPos

Horizontal Destination pixel position from left of Bitmap-0 to copy to.

yPos

Vertical Destination pixel position from top of Bitmap-0 to copy to.

xSize

Size of horizontal line in pixels (NOTE: Each pixel is 3 bytes: LSB=Red,Green,MSB=Blue) Used with Image pointer only.

ySize

Number of Lines. Used with Image pointer only.

Backgnd_ForegndN

Copy directly to currently displayed overlay image when 0. Otherwise copy to overlay's double buffered background.

handle

ID if multiple boards installed.

Returns

0 on success, -1 on failure.

```
MID2226_API int S2226_OverlayBackgroundColor(
   int red,
   int green,
   int blue,
   HANDLE hdev
);
```

Sets the color used for the background regions.

Parameters

red

reg component 0-255

green

green component 0-255

blue

blue component 0-255

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_MoveOverlay(
   int WinIndex,
   int xpos,
   int ypos,
   HANDLE hdev
);
```

Changes the position of an overlay sub-window

Parameters

WinIndex

index of window to move

xpos

new x position

ypos

new y position

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_OverlayDelXY(
   int xpos,
   int ypos,
   HANDLE hdev
);
```

Delete an overlay at x, y location if one exists at that location.

Parameters

xpos

x position

ypos

y position

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, -1 if overlay does not exist, -2 if that sub-window already used.

```
MID2226_API int S2226_OverlayDel(
   int WinIndex,
   HANDLE hdev
);
```

Delete overlay with index WinIndex.

Parameters

WinIndex

(0-7) sub-window position to query.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, -1 if overlay does not exist.

```
MID2226_API int S2226_UpdateOverlay(
   HANDLE hdev
);
```

Refresh or update the overlay to the hardware.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_ClearOverlay(
   HANDLE hdev
);
```

Clears all overlays for every region.

Parameters

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_ClearOverlayRegion(
   int regionmask,
   HANDLE hdev
);
```

Clears all overlays for specified region. Use S2226_ClearOverlay to erase overlays. This function only clears the overlay by region.

Parameters

regionmask

MID2226_REGION_MONITOR, MID2226_REGION_MPEG, MID2226_REGION_STILL.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetOverlayRegion(
   int regionmask,
```

```
   HANDLE hdev
);
```

Sets all overlays for specified region.

Parameters

regionmask

MID2226_REGION_MONITOR, MID2226_REGION_MPEG, MID2226_REGION_STILL.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetMergeMethod(
   MID2226_FIELDALG merge_method,
   void *unused,
   HANDLE hdev
);
```

Sets the interlaced field merging algorithm. Used for snapshots/stills.

Parameters

merge_method

MID2226_FIELDALG_NONE, MID2226_FIELDALG_DUP, MID2226_FIELDALG_MERGE,

MID2226_FIELDALG_INTER

unused

possible future use. Set to NULL.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SnapshotToFile(
   char *filename,
   int filetype,
```

```
   int freezetime,
   int wait,
   int qual,
   int unused,
   HANDLE hdev
);
```

Takes a snapshot and save to file in filename.

Parameters

filename

fully qualified file with path (without extension, extension will be added by the SDK).

filetype

file type to save. MID2226_FILE_JPEG and/or MID2226_FILE_BMP.

freezetime

time in milli-seconds to freeze the image(freezing is done on the video output channels of the 2226).

wait

wait = 1 will wait if operations pending(board is busy), wait = 0 will return err code if board busy.

unused

Unused. For future possible use. Value wil be ignore.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SnapshotToFileW(
   const wchar_t *filename,
   int filetype,
   int freezetime,
   int wait,
   int qual,
   int unused,
```

```
   HANDLE hdev
);
```

Takes a snapshot and save to file in filename (unicode version)

Parameters

filename

fully qualified unicode filename with path (without extension, extension will be added by the SDK).

filetype

file type to save. MID2226_FILE_JPEG and/or MID2226_FILE_BMP.

freezetime

time in milli-seconds to freeze the image(freezing is done on the video output channels of the 2226).

wait

wait = 1 will wait if operations pending(board is busy), wait = 0 will return err code if board busy.

unused

Unused. For future possible use. Value wil be ignore.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SnapshotToMem(
   BYTE *image,
   int size,
   int freezetime,
   int wait,
   HANDLE hdev
);
```

Get snapshot to memory function. Retrieves processed image into memery. Image is converted to RGB (1 byte per color) and the fields are merged using the algorithm set by S2226_SetMergeMethod.

Parameters

image

pointer to retrieved snapshot. (image must be preallocated).

size

size of the image buffer (image).

freezetime

time in milli-seconds to freeze the image(freezing is done on the video output channels of the 2226).

wait

wait = 1 will wait if operations pending(board is busy), wait = 0 will return err code if board busy.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SnapshotRaw(
   BYTE *image,
   int size,
   image_raw_t *image_raw,
   int freezetime,
   int wait,
   HANDLE hdev
);
```

Get snapshot data from hardware only. This function does no processing on the image, it just retrieves a pointer to the raw fields(field 1 will be null for 720p format) in YCrCb format. Image itself stored in image parameter. image_raw parameter used to indicate where the fields start and their size.

Parameters

image

pointer to pre-allocated space for image.

size

size of the image buffer (image).

image_raw

pointer to the returned raw image fields.

freezetime

time in milli-seconds to freeze the image(freezing is done on the video output channels of the 2226).

wait

wait = 1 will wait if operations pending(board is busy), wait = 0 will return err code if board busy.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetFirmwareVersions(
   unsigned int *fpga,
   unsigned int *usb,
   unsigned int *cpu,
   unsigned int *boardid,
   HANDLE hdev
);
```

Retrieves hardware firmware information. For support information.

Parameters

fpga

FPGA firmware version.

usb

USB firmware version.

CPU

Embedded CPU firmware version

boardid

Board revision identifier.

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetSDKVersions(
   unsigned int num,
   unsigned int *ver,
   HANDLE hdev
);
```

Retrieves version for SDK software. Useful for debugging Windows installation issues. See demo app for example usage.

Parameters

num

number of version to receive

ver

array of versions (driver core version[0], driver proxy version[1], dll version[2])

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_GetSN(
   SN2226 *sn,
   HANDLE hdev
);
```

Retrieves device serial number.

Parameters

sn

device serial number structure

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_SetOutput(
```

```
   int type,
   HANDLE hdev
);
```

Sets the output control for the device for the composite outputs.

Parameters

type

0-scaled output off(raw preview may be used), 1-scaled output on (raw preview not avaiiable).

hdev

handle to device (obtained from S2226_OpenBoard).

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_save_rgb(
   const char *fileName,
   const unsigned char *rgb,
   int width,
   int height,
   int type
);
```

This helper function does not involve the 2226 hardware. It is used in the snapshot demo program to save bitmaps from frames grabbed via the raw callback mechanism.

Parameters

fileName

full path to the target file in ASCII format, no extension.

rgb

pointer to the image in RGB format.

width

width of the image

height

height of the image

type

file save type. 0—BMP, 1—PPM.

```
MID2226_API int S2226_save_rgbW(
   const wchar_t *fileName,
   const unsigned char *rgb,
   int width,
   int height,
   int type
);
```

Unicode wide-char version of S2226_save_rgb.

Parameters

fileName

full path to the target file in Unicode format, no extension.

rgb

pointer to the image in RGB format.

width

width of the image

height

height of the image

type

file save type. 0—BMP, 1—PPM.

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

```
MID2226_API int S2226_uyvy_to_rgb(
   const unsigned char *uyvy,
   unsigned char *rgb,
   int width,
   int height
);
```

This helper function does not involve the 2226 hardware. It is used in the snapshot demo program to save bitmaps from frames grabbed via the raw callback mechanism. The frame captured via the

callback mechanism is usually UYVY. This converts it to RGB format before saving the image to file as BMP.

Parameters

uyvy

pointer to UYVY image. Size of this image is width * height * 2.

rgb

destination image address. pointer to resulting rgb image. Size must be at least width * height * 3.

width

width of the image

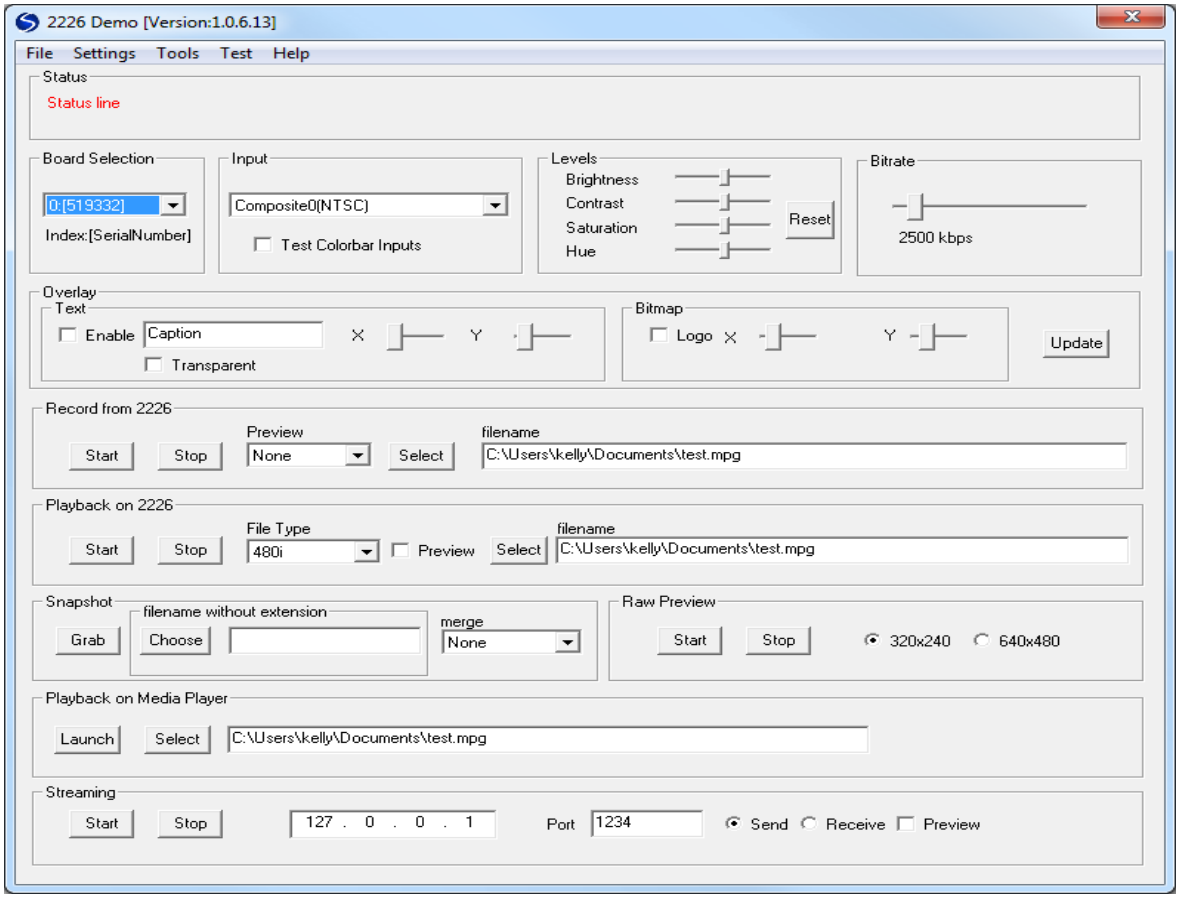
height

height of the image

Returns

0 on success, negative value if error (see mid2226types.h for error codes list).

Demo Application



The demo application is shown above.

Board Selection

The board selection allows the user to select between multiple boards in the system. The First number represents the zero indexed board number. The second number following the colon is the serial number.

The About button pops up a dialog showing the current board's firmware revisions.

Input

The input must be selected before performing any streaming operation. Select from the drop down. The "Colorbars" checkbox is used to internally generate colorbars on the 2226. Checking "Colorbars" adds the color bar inputs to the Input drop down. If color bars are desired, "Colorbars" must be selected AND the color bar input selection made from the drop down list. "2226TA" adds the auxiliary inputs

available using the 2226 termination board (2226TA). "2226S" is the default setting and displays the inputs on the boxed 2226 system.

Levels

The brightness, contrast, hue and saturation may be changed at any time with the slider bars.

Bitrate

Bitrate is the encoded bitrate used when streaming and recording from the 2226.

Overlay

Demonstrates some of the 2226 overlay capability. Click the Update button to refresh the current overlay settings on the 2226.

Record from 2226

Record from the 2226 to file. Files are encoded as H.264 MPEG with the selected bitrate. A raw preview window is displayed if the "raw" option is selected from the preview dropdown box. A decoded preview window is displayed if the "Decoded" option is selected from the preview dropdown box.

Playback on 2226

Decodes the recorded H264 stream on the 2226 and output to hardware video outputs. Please make sure the file type matches the input before starting. A raw preview window is displayed if the "Preview" checkbox is selected.

Snapshot

Grabs an snapshot from the 2226. When the file dialog appears, do not enter an extension. A bitmap and JPG file will be created with the required file extensions.

Raw Preview

A raw preview window, at the selected resolution, is started when "Start" button is pressed. The raw preview window is removed when the "Stop" button is pressed.

Playback on Media Player

Launches the selected file on the default media player. WMP support for Win7 only. If running on XP, the demo will check for the VLC player.

Streaming

Streams the encoded H264 transport stream over UDP.

File/Exit

Closes and exits the Demo Application.

Settings/Audio Routing...

Audio settings brings up the audio gain and output settings.

Settings/Mpeg-In

Selects which audio input is selected to drive the input to the Mpeg encoder.

Settings/Line-Out

Selects which audio input is selected to drive the input to the Line-out driver.

Settings/SDI-Out

Selects which audio input is selected to be encoded into the SDI-Out video stream.

Settings/Audio Config

Audio settings brings up the audio gain and output settings.

Settings/Scaled Composite Output

This is a legacy control for scaling the raw preview or the composite output, before the 2226 hardware could do both.

Settings/Adapter Board

When selected, this option adds to the input selection box all the additional video inputs provided on the video header and connected to BNC connecters with the optional adapter board.

Tools/Audio Meter

This option toggles whether the Audio Meter is displayed or not.

Tools/TCL Scripting

This option opens a command window running a TCL interpreter that has most of the 2226 API routines attached to TCL commands. See appendix A.

Test/Overlay Test

This option creates a new thread that continuously generates moving text and images. When selected a second time, the thread is stopped and deleted.

Help/About

This option opens a Dialog box documenting the Hardware and software version numbers.

Audio Meter/Left,Right/Level

The hexadecimal values returned from a call to S2226_GetAudioMtrLevel().

Audio Meter/Left,Right/dB

The values returned from a call to S2226_GetAudioMtr_dB().

Audio Meter/Left,Right/Hold dB

The values returned from a call to S2226_GetAudioMtrHold().

Audio Meter/Settings/Test

Actually the Input Selection. Selects the Audio input that feeds the Meter.

Audio Meter/Settings/Input

Actually the Test Selection. Selects whether a test tone is generated and at what level.

Audio Meter/Hold/Selection

Selects the time duration that a maximum amplitude is held on the hold register.

Audio Meter/Hold/Release

When checked, allows the Hold circuit to release the current maximum value and hold the next highest value after the hold duration.

When not selected, the maximum value in the hold register will not change unless a new maximum is detected.

Snapshot Demo Application

The snapshot demo application was designed for customers that need to retrieve and save images faster than allowed by the S2226_SnapshotToFile function. Instead of handshaking with the device, the raw preview data is streamed and captured via callback. The user is responsible for ensuring that an allowed scaling is performed.

The snapshot demo can be launched from the Start menu. Start->Program Files->Sensoray->2226->2226 Snapshot Demo.

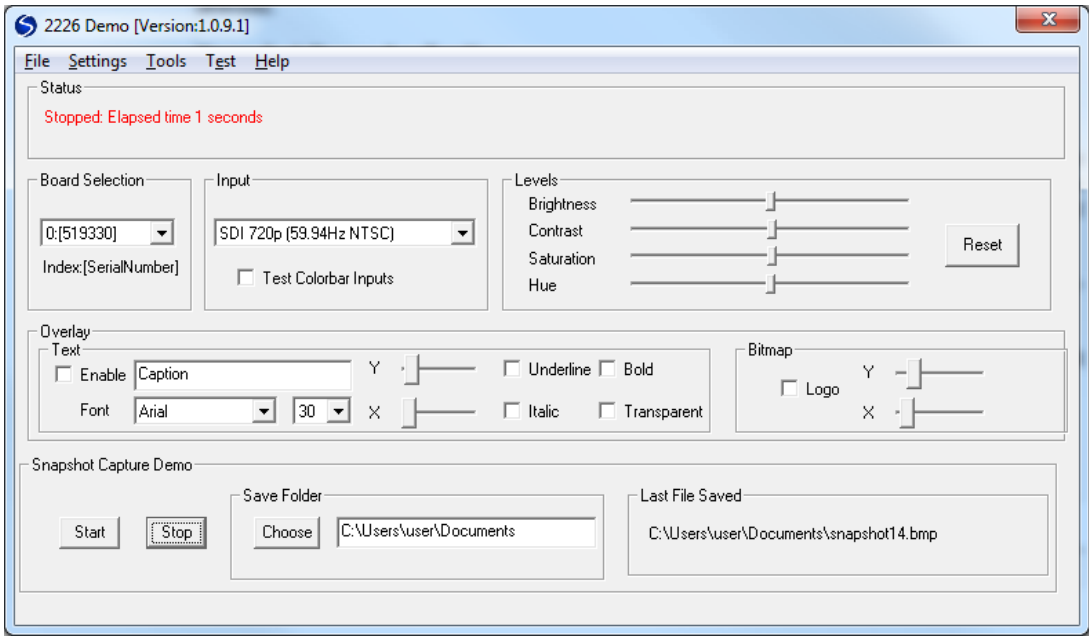
Due to USB2.0 bandwidth limitations, any scaling over 640x480 will not be at full frame rate. At Sensoray, we measured 10fps for 1280x720 and 4fps for 1920x1080. Results may vary.

The allowed preview scales using S2226_SetRawPreviewSize are as follows:

SDI-SD, Composite, Svideo: 320x240, 640x480, 1280x720 (reduced frame rate), 1920x1080

SDI-HD (720): 320x240, 640x480, 1280x720

SDI-HD(1080): 320x240, 640x480, 1920x1080



To use the snapshot, first connect and power on your camera source to the 2226. In the demo, select the corresponding input. The next step is optional. If desired, you may change the target output directory. Once this is done, you may click "Start" and snapshots will be captured.

Internally, the snapshot demo uses the following functions. S2226_SetInput, S2226_RegisterCallbackRaw, S2226_SetRawPreviewSize, S2226_StartRawPreviewCallback, S2226_StopRawPreview.

FAQ

Q1) Can the 2226 record one file and decode another at the same time?

A1) The 2226 is half duplex. Only one operation (encode/decode) may be performed at a time.

Q2) I need to pause the video. Why does the pause button in GraphEdit not work?

A2) The hardware does not support pausing of the stream. If pause is required, it may be possible to construct a custom callback function to drop packets. At this time, Sensoray does not provide support for such a feature. The recommended way to support pause is to record with separate clips.

Q3) I can't play back the recorded file in Windows media player on XP?

A3) XP and Vista do not necessarily have the H264 decoders to decode the stream. The recorded stream may be played back with WMP under Windows 7. A player that works under XP is VideoLan.

Q4) What are the adapter inputs in the demo application?

A4) An optional termination board is available for the 2226 (2226-TA) to provide S-Video and an extra composite video channel. This does not apply for the boxed 2226S unit.

Q5) Why do the image snapshots show 2 images.

A5) The S2226_SnapshotToFile and S2226_SnapshotToFileW functions work in conjunction with S2226_SetMergeType. For maximum flexibility in the API, the user is given full control over how the fields are presented. Change the merge type in the demo application to merge the fields into one frame.

Appendix A - TCL Scripting

The 2226 Demo application includes a TCL scripting language interface. It is accessed from the Tools menu. The version used is based on Jim Tcl. Jim Tcl is a small footprint reimplementation of the Tcl scripting language. The core language engine is compatible with Tcl 8.5+, while implementing a significant subset of the Tcl 8.6 command set, plus additional features available only in Jim Tcl.

Tcl stands for tool command language and is pronounced tickle. It is actually two things: a language and a library.

First, Tcl is a simple textual language, intended primarily for issuing commands to interactive programs such as text editors, debuggers, illustrators, and shells. It has a simple syntax and is also programmable, so Tcl users can write command procedures to provide more powerful commands than those in the built-in set.

Second, Tcl is a library package that can be embedded in application programs. The Tcl library consists of a parser for the Tcl language, routines to implement the Tcl built-in commands, and procedures that allow each application to extend Tcl with additional commands specific to that application. The application program generates Tcl commands and passes them to the Tcl parser for execution. Commands may be generated by reading characters from an input source, or by associating command strings with elements of the application's user interface, such as menu entries, buttons, or keystrokes.

When the Tcl library receives commands it parses them into component fields and executes built-in commands directly. For commands implemented by the application, Tcl calls back to the application to execute the commands. In many cases commands will invoke recursive invocations of the Tcl interpreter by passing in additional strings to execute (procedures, looping commands, and conditional commands all work in this way).

An application program gains three advantages by using Tcl for its command language. First, Tcl provides a standard syntax: once users know Tcl, they will be able to issue commands easily to any Tcl-based application. Second, Tcl provides programmability. All a Tcl application needs to do is to implement a few application-specific low-level commands. Tcl provides many utility commands plus a general programming interface for building up complex command procedures. By using Tcl, applications need not re-implement these features.

Third, Tcl can be used as a common language for communicating between applications. Inter-application communication is not built into the Tcl core described here, but various add-on libraries, such as the Tk toolkit, allow applications to issue commands to each other. This makes it possible for applications to work together in much more powerful ways than was previously possible.

Fourth, Jim Tcl includes a command processor, jimsh, which can be used to run standalone Tcl scripts, or to run Tcl commands interactively.

For detailed information on the TCL language see:

http://jim.tcl.tk/fossil/doc/trunk/Tcl_shipped.html

Using TCL

Upon selecting TCL from the Tools menu, an interactive (console) environment is setup for running and creating TCL scripts.

Command can be executed one at a time, saved in a file and called.

One such script is Demo.tcl located in C:\...install_path..\Sensoray\2226\Tcl

This script uses new windows GDI wrapper routines to implement a full screen overlay API that is from 20 times faster than the original Overlay API. Speedup varies depending on the actual number of pixels contained in the overlay and the number of pixels that need to be updated.

Procedures and command execution

The user is free to run commands immediately or to create new procedures for later execution.

Procedures are defined as:

```
proc welcome { param } {
 puts " Hello $param"
}
```

and can be executed by entering the procedure name, the parameter to use, and hitting enter:

```
welcome John
```

Produces:

```
Hello John
```

External scripts in files can be run by using the source command:

```
source ..\\..\\tcl\\demo.tcl
source ../../tcl/demo.tcl
```

Many of the 2226's API routines have corresponding TCL commands. See Appendix B and C for more detailed descriptions of these.

The user is encouraged to wrap more API routines as well as their own routines into TCL commands. This is easily done by modifying jim_TclShell.cpp using the existing commands as templates.

Integration with the 2226

Integration with the 2226 is done in a single file:

```
jim_TclShell.cpp
```

The routines that wrap the API function's typically come in three flavors. Interactive, Self-Contained, and Interdependent.

Interactive

The most common routines are the interactive routines that parse parameters that are passed in, make the API call and then return a value.

Self-Contained

Some of the routines, like tcl_gtest, are self-contained. They take no TCL parameters, and always return successfully.

Interdependent

The last type of routines are Interdependent. These include the fast graphics Windows GDI to Overlay wrappers. They create a simple environment that can produce sophisticated full graphics that can be quickly generated and updated on the fly.

For information on writing TCL scripts see example scripts located in the TCL directory, and the following web sites.

http://jim.tcl.tk/fossil/doc/trunk/Tcl_shipped.html

http://www.tcl.tk/man/tcl8.5/

Appendix B – API wrappers

The following 2226 API routines have been wrapped, and can be called, with TCL commands.

Settings

Reducedclock / rc n

Calls S2226_SetReducedClock()

This routine prepares the input function to set the video frame-rate / clock rate to 60 Hz or 59.9 Hz.

Valid choices for n:

0=60Hz, 1=59.94Hz

vidsys s

Calls S2226_SetVidSys()

This routine prepares the input function for NTSC or PAL when a composite, s-video, or SD-SDI source is selected.

Valid choices for s:

2-NTSC, 1-PAL

get_vidsys

Calls S2226_GetVidSys()

This routine returns the vidsys number given to the vidsys command.

input i

Calls S2226_SetInput()

This routine selects the input source and resolution. For the HD-SDI input's the clock rate, 59Hz or 60 Hz, must be selected with the rc or reducedclock function prior to calling this.

Valid choices are I:

0-comp0, 1-svid0, 2-comp1, 3-svid1, 4-CB, 5-720pCB, 6-1080iCB, 7-SDI-SD, 8-SDI-720P, 9-SDI-1080i

get_input

Calls S2226_GetInput()

This routine returns the input number given to the input command.

outputscale type

Calls S2226_SetOutput()

Sets the output control for the device for the composite outputs. Multiplexed with the raw preview feature. type-0=unscaled, type-1=scaled.

setbright n

Calls S2226_SetLevel(MID2226_LEVEL_BRIGHTNESS,...)

Sets brightness of the captured video to n.

setcont n

Calls S2226_SetLevel(MID2226_LEVEL_CONTRAST,...)

Sets contrast of the captured video to n.

setsat_cr n

Calls S2226_SetLevel(MID2226_LEVEL_HUE,...)

Sets hue of the captured video to n.

setsat_cb n

Calls S2226_SetLevel(MID2226_LEVEL_SATURATION,...)

Sets saturation of the captured video to n.

Record / Playback

record filename

Calls S2226_StartRecord()

This routine starts recording the selected input to the given file-filename.

record_preview_raw filename

This routine calls S2226_SetPreviewType() and S2226_StartPreviewAndRecord() to record the selected input to the file-filename and to display what is being recorded on the computer monitor using the MID2226_PREVIEWTYPE_RAW method. This uses the least number of CPU cycles since the video being displayed is not compressed.

record_preview_decoded filename

This routine calls S2226_SetPreviewType() and S2226_StartPreviewAndRecord() to record the selected input to the file-filename and to display what is being recorded on the computer monitor using the MID2226_PREVIEWTYPE_DECODED method. This uses the most number of CPU cycles since the video being displayed must be decompressed before it is displayed.

preview_decoded

This routine calls S2226_SetPreviewType() and S2226_StartPreview() to display what is being being recorded or payed. It uses the MID2226_MID2226_PREVIEWTYPE_DECODED_RAW method to display the video.

preview_raw

This routine calls S2226_SetPreviewType() and S2226_StartPreview() to display what is being being recorded or payed. It uses the MID2226_PREVIEWTYPE_RAW method to display the video.

play format

This routine calls S2226_StartDecode() to decode a given compressed file on the monitors driven by the 2226. The format parameter can be 0-7 or 480i, 576i, 720p@50, 720p@59, 720p@60, 1080i@50, 1080i@59, 1080i@60.

stream ip port

This routine calls S2226_RegisterCallback() and S2226_StartCallback() to start recording and sending recorded output to ip address-ip on UDP port-port.

stop

Calls S2226_StopStream()

This routine stops recording or playing mode.

Snapshot

Setmerge m

Calls S2226_SetMergeMethod()

This routine sets the method that an interlaced video snapshot is de-interlaced before being saved to file. The merge type-m can be: 0=None, 1=Duplicate field 0 twice, 2=Merge field 0 and filed 1, 3=Interpolate field 0 lines to create a second field to interlace.

Snapshot f t z

Calls S2226_SnapshotToFile()

This routine freezes the display for z takes a snapshot of the current input video selected, de-interlaces it using the routine set by SetMergeMethod(), then saves the results to the filename-f in the filetypes set with type-t.

Overlay

Ovltext t x y p

This routine calls S2226_OverlayText() and optionally S2226_UpdateOverlay()

It adds ARIAL font, white text-t on a transparent background to the next available overlay index, for the output Monitor, Snapshot, and Recorded video streams using the point size-p at the given x,y location.

S2226_UpdateOverlay() is called if the global variable 'autoupdate' has been set with the autoupdate command.

ovltexti t i x y p r g b R G B

This routine calls S2226_OverlayTextIdx() and optionally S2226_UpdateOverlay()

It adds text-t to the overlay using index-i, pointsize-p, text color-r-g-b, background color-R-G-B, at location x,y.

Ovlbackcolor r g b

Calls S2226_OverlayBackgroundColor()

This routine sets the color used for the background regions

ovlimage f r i x y

Calls S2226_OverlayImageIdx() and optionally S2226_UpdateOverlay()

This routine overlays .bmp file f to regions r, at x,y location using overlay index-i.

ovlimagei f i x y

Calls S2226_OverlayImageIdx() and optionally S2226_UpdateOverlay()

Overlay .bmp file f to location x, y using index i

ovlimageraw p xSz ySz i x y

Calls S2226_OverlayImageRaw() and optionally S2226_UpdateOverlay()

Put image p of xSz ySz to overlay index-i at x,y

ovlimagetest i x y

This routine calls the S2226_OverlayImageRaw() API with a known hardcoded image, puts the image to overlay index-i at x,y

ovlblitf f x y g

Calls S2226_CopyBmpToOverlayZero()

This routine will copy Overlay .bmp file f to loc x, y within Overlay Index-0. If g=0=foreground. Image will appear immediately. If g=1=background. Image will be written to background and will appear after bankswitch initiated by S2226_UpdateOverlay()

ovlblit p xSz ySz x y [g=0]

Calls S2226_CopyBmpToOverlayZero()

This routine copys overlay image p size xSz,ySz to loc x, y within Overlay Index-0. If g=0=foreground. Image will appear immediately. If g=1=background. Image will be written to background and will appear after bankswitch=S2226_UpdateOverlay()

Item g is optional. If not given it will default to 0.

ovlmove i x y

Calls S2226_MoveOverlay() and optionally S2226_UpdateOverlay()

This routine move overlay sub-window-i to new x,y location

ovldelxy x y

Calls S2226_OverlayDelXY() and optionally S2226_UpdateOverlay()

This routine deletes the overlay index at location x y.

ovldel i

Calls S2226_OverlayDel() and optionally S2226_UpdateOverlay()

This routine deletes overlay index-i

ovlclear

Calls S2226_ClearOverlay()

This routine clears all overlay index locations.

ovllist

This routine calls S2226_GetOverlayIdx() repeatedly to print a list of all overlay items.

ovlupdate

Calls S2226_UpdateOverlay()

This routine merges and clips all overlay index windows, downloads them to shadow overlay memory in the 2226, and then bank-switches / swaps the shadow memory with the displayed overlay memory at then next top of frame.

Miscellaneous

version

Calls S2226_GetFirmwareVersions()

This routine returns string pointers to Middleware Version, Fpga Version, and Board Version and then then prints them to the console.

Utility

autoupdate

This routine sets a global variable that many of the routines check to see whether to automatically call S2226_UpdateOverlay() after each routine. When clear, many ovlxxx API calls can be made before calling S2226_UpdateOverlay() manually to flush the new changes to the screen.

debugtime

This routine sets a global variable that causes many of the routines to measure the time it takes to call the API routines, and then print the results.

kbhit

This routine senses whether the user presses a key on the keyboard. It can be used to break out of infinite loops.

hexpr

This routine, written in Tcl itself, calls the Tcl expr functionand returns the result in hexidecimal prefixed with "0x". This result can then be used as a number in further computations.

Appendix C - Fast graphic overlays

The new fast graphics Overlay API consists of a single function:

```
SN_CopyBmpToOverlayZero(...)
```

This function can be hard to use, but is made much easier with the supporting TCL gxxx routines. (gopen, gclose, gwrite, gpen, gbrush, gsetrop2, gbkmode, gsetbkcolor, gtextcolor, gfont, gtext, grect, grrect, gellipse, gmoveto, glineto, gloadimage, gbitblit, gstretchblt, gfixblack, gtest)

The heart of the new framework is the gwrite command which calls tcl_gwrite() in jim_TclShell.cpp

In this procedure, a Windows GDI bitmap and device context pair is converted to a flat RGB bitmap that is then sent to the 2226 hardware via the SN_CopyBmpToOverlayZero() API routine. The advantage of this API routine over the other 2226 API overlay routines is that it only downloads the pixels in the memory structure passed to it. The other API routines must merge all eight window-index regions and download the combined count of pixels.

In addition to this fast update rate, this routine is more flexible in that the source of the overlay to be downloaded is a Windows bitmap and all the Windows GDI tools can be used to create it's contents on the fly.

The fast graphics tools are divided in to Support, Graphics, Text, and Bitmap routines.

Support

screen_size

This routine is a TCL procedure that other TCL routines can use to determine the Width and Height of the Video input. It returns the input video Width and Height in a list.

cls

This routine is a TCL procedure that is used to generate and download a full-screen transparent overlay to Overlay-Index 0.

Calling this function prior to using the gxxx routines satisfies CopyBmpToOverlayZero()'s Width and Height restrictions.

This could also be done by using ovlimagei to load a full screen image to Index-0 (i.e. ovlimagei [fname $imagpath HD_Halo_with_hole.bmp] 0 0 0) Where the file HD_Halo_with_hole.bmp is the same size as the selected input resolution.

gtest

This routine is a self-contained example of how to use the Windows GDI graphics routines with the fast CopyBmpToOverlayZero() API routine. It creates a Device Context and Selected Bitmap for creating graphics. Draws some graphics onto this Bitmap. Converts this Bitmap to a flat RGB bitmap that can be downloaded. Downloads the bitmap via CopyBmpToOverlayZero() and then cleans up/frees the drawing resources that were used.

The rest of the gxxx routines, follow the same sequence as define in this routine, but break it up into modular pieces that can be used to build dynamic overlays.

gopen xsz ysz

The purpose of gopen is to create the Handle to Device Context, and Handle to Bitmap of size xsz by ysz, that Windows needs to write graphics into. The key to this is creating the Compatible Bitmap, g_hbmMem, from a Device Context generated from the Desktop, not the Memory Device Context (that is also created from the Desktop.) This is because CreateCompatibleDC() will be given a default 1x1 monochrome bitmap that subsequent SelectBitmap() statements will not change the color of. i.e. it will remain monochrome.

The other function of gopen is to setup the context's pen, brush, background color, font, and font color with default values that may have been changed with other gxxx functions. This allows gopen to be called multiple times without having to re-program all the default values.

NOTE: This routine, and all the gxxx routines, assume that cls has been called first, or that a suitable full screen overlay has been loaded into overlay index-0 with the ovlimage, ovlimagei, or ovlimageraw statements.

gclose

This routine releases all the handles to pens, brushes, bitmaps, etc.. that the gxxx tools use to create graphics.

gwrite x y [BackFore_n=0]

The routines gets the bitmap, that is being drawn on, in the memory Device Context, converts it to a format compatible for download, and then downloads the pixels in this overlay to the boards overlay memory at location x,y using SN_CopyBmpToOverlayZero() API so only the pixels in the bitmap need to be sent to the board. The default is to load the overlay ontop of the overlay currently being displayed, thus no call to S2226_UpdateOverlay() is needed to display it.

gfixblack [x=dc_W] [y=dc_H]

Since the 2226 uses 16-bit color for the overlays, and since it treats 0,0,0 as a special transparent color, values in the range from 0,0,0 to 3,3,3 will inadvertently be rounded down to the 0,0,0 transparent color.

This routine searches the bitmap for RGB values from 0,0,0 to 3,3,3 and replaces them with 4,4,4. This is useful because the Text anti-aliasing can create these color values when rendering text to the bitmap.

The optional x,y parameters determine the region within the device context/bitmap to perform the convertion. If not given, the routine will convert the whole bitmap.

Graphics

gpen t w r g b

This is a simple wrapper function for the Windows CreatePen () function. It also sets a global default values for subsequent calls to gopen to set the new device context/bitmap with the same pen.

T = type. Valid values are 0-6.

```
PS_SOLID            0
PS_DASH             1       /* -------  */
PS_DOT              2       /* .......  */
PS_DASHDOT          3       /* _._._._  */
PS_DASHDOTDOT       4       /* _.._.._  */
PS_NULL             5
PS_INSIDEFRAME      6
```

w = width.

r = red. Valid values are 0-255.

g = green. Valid values are 0-255.

b = blue. Valid values are 0-255.

gbrush r g b

This is a simple wrapper function for the Windows CreateSolidBrush () function. It also sets a global default value for subsequent calls to gopen.

It creates a solid brush with color:

r = red. Valid values are 0-255.

g = green. Valid values are 0-255.

b = blue. Valid values are 0-255.

gsetrop2 op

This is a simple wrapper function for the Windows SetROP2 () function.

The op parameter controls how the bitmap is copied.

Valid op values:

```
R2_BLACK            1   /*  0       */
R2_NOTMERGEPEN      2   /* DPon     */
R2_MASKNOTPEN       3   /* DPna     */
R2_NOTCOPYPEN       4   /* PN       */
R2_MASKPENNOT       5   /* PDna     */
R2_NOT              6   /* Dn       */
R2_XORPEN           7   /* DPx      */
R2_NOTMASKPEN       8   /* DPan     */
R2_MASKPEN          9   /* DPa      */
R2_NOTXORPEN        10  /* DPxn     */
R2_NOP              11  /* D        */
R2_MERGENOTPEN      12  /* DPno     */
R2_COPYPEN          13  /* P        */
R2_MERGEPENNOT      14  /* PDno     */
R2_MERGEPEN         15  /* DPo      */
R2_WHITE            16  /*  1       */
R2_LAST             16
```

gbkmode mode

This is a simple wrapper function for the Windows SetBkMode () function.

Valid values for mode:

1=TRANSPARENT

2=OPAQUE

gsetbkcolor r g b

This is a simple wrapper function for the Windows SetBkColor () function. It also sets a global default value for subsequent calls to gopen.

Valid values for the color are:

r = red. Valid values are 0-255.

g = green. Valid values are 0-255.

b = blue. Valid values are 0-255.

grect xL yT xR yB

This is a simple wrapper function for the Windows Rectangle () function.

Arguments:

xL = Horizontal coordinate for the top left pixel of text in device context / bitmap

yT = Vertical coordinate for the top left pixel of text in device context / bitmap

xR = Horizontal coordinate for the bottom right pixel of text in device context / bitmap plus one.

YB = Vertical coordinate for the bottom right pixel of text in device context / bitmap plus one.

grrect xL yT xR yB w h

This is a simple wrapper function for the Windows RoundRect () function.

Arguments:

xL = Horizontal coordinate for the top left pixel of text in device context / bitmap

yT = Vertical coordinate for the top left pixel of text in device context / bitmap

xR = Horizontal coordinate for the bottom right pixel of text in device context / bitmap plus one.

YB = Vertical coordinate for the bottom right pixel of text in device context / bitmap plus one.

w = Round corners in horizontal direction by this amount.

H = Round corners in vertical direction by this amount.

gellipse xL yT xR yB

This is a simple wrapper function for the Windows Ellipse () function.

Arguments:

xL = Horizontal coordinate for the top left pixel of text in device context / bitmap

yT = Vertical coordinate for the top left pixel of text in device context / bitmap

xR = Horizontal coordinate for the bottom right pixel of text in device context / bitmap plus one.

YB = Vertical coordinate for the bottom right pixel of text in device context / bitmap plus one.

gmoveto x y

This is a simple wrapper function for the Windows MoveToEx () function. Moves a virtual cursor without drawing anything.

Arguments:

x = Horizontal coordinate for the starting pixel in device context / bitmap

y = Vertical coordinate for the starting pixel in device context / bitmap

glineto x y

This is a simple wrapper function for the Windows LineTo () function. Draws a line from the current virtual cursor position to the given coordinate.

Arguments:

x = Horizontal coordinate for the end point of a line in device context / bitmap

y = Vertical coordinate for the end point of a line in device context / bitmap

Text

gtextcolor r g b

This is a simple wrapper function for the Windows SetTextColor () function. It also sets a global default value for subsequent calls to gopen.

Valid values for the color are:

r = red. Valid values are 0-255.

g = green. Valid values are 0-255.

b = blue. Valid values are 0-255.

gfont f h [wi=0][p=0][we=0][i=0][u=0][s=0][es=0][or=0]

This is a simple wrapper function for the Windows CreateFont () function. It also sets a global default value for subsequent calls to gopen.

Valid values:

f = font name

h = height

wi = width

p = pitch (0,1,2,8)

DEFAULT_PITCH	0
FIXED_PITCH	1
VARIABLE_PITCH	2
MONO_FONT	8

we = weight (0,100-900)

FW_DONTCARE	0
FW_THIN	100
FW_EXTRALIGHT	200
FW_LIGHT	300
FW_NORMAL	400
FW_MEDIUM	500
FW_SEMIBOLD	600
FW_BOLD	700
FW_EXTRABOLD	800
FW_HEAVY	900

i = italic (0,1)

u = underline (0,1)

s = strikeout (0,1)

esc = escapement

or = orientation

gtext t xL yT xR yB [s=0]

This is a simple wrapper function for the Windows DrawText () function. It also returns Width and Height values when DrawText() is called with the DT_CALCRECT option for calculating the bounding rectangle for the given text.

The arguments are:

t = text to be overlayed

xL= Horizontal coordinate for the top left pixel of text in device context / bitmap

yT = Vertical coordinate for the top left pixel of text in device context / bitmap

xR= Horizontal coordinate for the bottom right pixel of text in device context / bitmap

yB = Vertical coordinate for the bottom right pixel of text in device context / bitmap

s = style

DT_TOP	0x00000000
DT_LEFT	0x00000000
DT_CENTER	0x00000001
DT_RIGHT	0x00000002
DT_VCENTER	0x00000004
DT_BOTTOM	0x00000008
DT_WORDBREAK	0x00000010
DT_SINGLELINE	0x00000020
DT_EXPANDTABS	0x00000040
DT_TABSTOP	0x00000080
DT_NOCLIP	0x00000100
DT_EXTERNALLEADING	0x00000200
DT_CALCRECT	0x00000400
DT_NOPREFIX	0x00000800
DT_INTERNAL	0x00001000
DT_EDITCONTROL	0x00002000
DT_PATH_ELLIPSIS	0x00004000
DT_END_ELLIPSIS	0x00008000
DT_MODIFYSTRING	0x00010000
DT_RTLREADING	0x00020000
DT_WORD_ELLIPSIS	0x00040000
DT_NOFULLWIDTHCHARBREAK	0x00080000
DT_HIDEPREFIX	0x00100000
DT_PREFIXONLY	0x00200000

Bitmap

gloadimage f

This is a wrapper function for the Windows LoadImage () function.

It loads the image into a second Bitmap/Device Context pair, g_hbmLoad and g_hdcLoad, respectively.

This second Bitmap/Device Context can then be used to copy the loaded image to the main Bitmap/Device Context multiple times. (see gbitblit and gstretchblit)

The Width and Height of the loaded image are returned as the result of this function.

Arguments:

f = Name and path of file to load.

gbitblit xDest yDest w h xSrc ySrc Rop

This is a simple wrapper function for the Windows BitBlt () function. The source parameter for the BitBlit() function is g_hdcLoad. This Device context, (and associated Bitmap), are loaded using the gloadimage function.

This function can be used to write the loaded image to the destination bitmap multiple times for shifting or rotation type of effects. See Demo.tcl

Arguments:

xDest, yDest = Destination coordinates in device context / bitmap created with gopen.

w h = Width and Height of bitmap to copy. (in pixels)

xSrc ySrc = Source coordinates in secondary device context / bitmap created with gloadimage.

Rop = Raster bit manipulation operation for copying the bitmap.

```
SRCCOPY             (DWORD)0x00CC0020 /* dest = source                   */
SRCPAINT            (DWORD)0x00EE0086 /* dest = source OR dest           */
SRCAND              (DWORD)0x008800C6 /* dest = source AND dest          */
SRCINVERT           (DWORD)0x00660046 /* dest = source XOR dest          */
SRCERASE            (DWORD)0x00440328 /* dest = source AND (NOT dest )   */
NOTSRCCOPY          (DWORD)0x00330008 /* dest = (NOT source)             */
NOTSRCERASE         (DWORD)0x001100A6 /* dest = (NOT src) AND (NOT dest) */
MERGECOPY           (DWORD)0x00C000CA /* dest = (source AND pattern)     */
MERGEPAINT          (DWORD)0x00BB0226 /* dest = (NOT source) OR dest     */
PATCOPY             (DWORD)0x00F00021 /* dest = pattern                  */
PATPAINT            (DWORD)0x00FB0A09 /* dest = DPSnoo                   */
PATINVERT           (DWORD)0x005A0049 /* dest = pattern XOR dest         */
DSTINVERT           (DWORD)0x00550009 /* dest = (NOT dest)               */
BLACKNESS           (DWORD)0x00000042 /* dest = BLACK                    */
WHITENESS           (DWORD)0x00FF0062 /* dest = WHITE                    */
NOMIRRORBITMAP      (DWORD)0x80000000 /* Do not Mirror the bitmap in this call */
CAPTUREBLT          (DWORD)0x40000000 /* Include layered windows */
```

gstretchblt xDest yDest wDest hDest xSrc ySrc wSrc hSrc Rop

This is a simple wrapper function for the Windows BitBlt () function. The source parameter for the BitBlit() function is g_hdcLoad. This Device context, (and associated Bitmap), are loaded using the gloadimage function.

Arguments:

xDest, yDest = Destination coordinates in device context / bitmap created with gopen.

wDest, hDest = Width and Height of destination bitmap after copy. (in pixels)

xSrc, ySrc = Source coordinates in secondary device context / bitmap created with gloadimage.

wSrc, hSrc = Width and Height of source bitmap to copy. (in pixels)

Rop = Raster bit manipulation operation for copying the bitmap.

```
SRCCOPY             (DWORD)0x00CC0020 /* dest = source                    */
SRCPAINT            (DWORD)0x00EE0086 /* dest = source OR dest            */
SRCAND              (DWORD)0x008800C6 /* dest = source AND dest           */
SRCINVERT           (DWORD)0x00660046 /* dest = source XOR dest           */
SRCERASE            (DWORD)0x00440328 /* dest = source AND (NOT dest )    */
NOTSRCCOPY          (DWORD)0x00330008 /* dest = (NOT source)              */
NOTSRCERASE         (DWORD)0x001100A6 /* dest = (NOT src) AND (NOT dest) */
MERGECOPY           (DWORD)0x00C000CA /* dest = (source AND pattern)      */
MERGEPAINT          (DWORD)0x00BB0226 /* dest = (NOT source) OR dest      */
PATCOPY             (DWORD)0x00F00021 /* dest = pattern                   */
PATPAINT            (DWORD)0x00FB0A09 /* dest = DPSnoo                    */
PATINVERT           (DWORD)0x005A0049 /* dest = pattern XOR dest          */
DSTINVERT           (DWORD)0x00550009 /* dest = (NOT dest)                */
BLACKNESS           (DWORD)0x00000042 /* dest = BLACK                     */
WHITENESS           (DWORD)0x00FF0062 /* dest = WHITE                     */
NOMIRRORBITMAP      (DWORD)0x80000000 /* Do not Mirror the bitmap in this call */
CAPTUREBLT          (DWORD)0x40000000 /* Include layered windows */
```