PC/104+ MPEG Capture Device

Model 314 (Rev.A)

Windows SDK User's Manual

July, 2006

7313 SW Tech Center Dr.
Tigard, OR 97223
Phone 503.684.8073 • Fax 503.684.8164
sales@sensoray.com
www.sensoray.com

Table of Contents

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.

Special handling instructions

The circuit board contains CMOS circuitry that is sensitive to Electrostatic Discharge (ESD).

Special care should be taken in handling, transporting, and installing circuit board to prevent ESD damage to the board. In particular:

- Do not remove the circuit board from its protective anti-static bag until you are ready to install the board into the enclosure.
- Handle the circuit board only at grounded, ESD protected stations.
- Remove power from the equipment before installing or removing the circuit board.

Introduction

Model 314 is a PC/104+ video and audio capture device. It converts the signal from an analog video source (NTSC or PAL) into one of the supported MPEG or MJPEG streams along with (optional) audio from a line or microphone input. Model 314 is powered through the PC/104+ bus and does not require any external power supplies.

Feature Summary

- MPEG1/2/4 or MJPEG capture at full frame rate and full resolution.
- 4 Composite or 2 S-video inputs.
- Stereo audio is mixed synchronously into MPEG stream (mono for the microphone input).
- On screen display (OSD) – multiple text overlays.
- Three-zone motion detection.
- Free Windows drivers, demo application and a Software Development Kit (SDK).

Specifications

Inputs	Video: composite 1-4 (BNC), S-video 1-2(DIN), 75 Ohm Audio: line in stereo (2xRCA), 10 kOhm.
Electret microphone bias current, max	3mA
Input video formats	NTSC (M), PAL (BDGHIMN)
Output formats	MPEG1, MPEG2 (MP@ML), MPEG4 (SP@L3 with B-frames), MJPEG (Motion JPEG)
Output resolutions	720x480 (NTSC, 30 frames/sec), 720x576 (PAL, 25 frames/sec), 352x240 (SIF), 352x288 (CIF)
OSD	max 96 characters per frame, 16x16 font
Snapshot	Raw, JPG and BMP formats, concurrent with capture/preview
Power	+5V, 500mA (through PC/104+ bus)

Software

Feature Summary

Model 314 is shipped with drivers for Microsoft Windows 2000/XP. A full-featured demo application allows preview and saving of MPEG video on the host computer, control of the compression and video settings, text overlays, and snapshot capture.

An included SDK can be used to integrate video capture into any application. The SDK allows maximum flexibility by providing an API for all the 314's functions. The source code of the demo application is a suggested starting point for custom application development.

Installation

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

Run the setup program (314setup.exe) from the distribution disk or folder. Software components, including a demo application with the source code, will be installed into the /Program Files/Sensoray/314 folder.

During the installation the program will install ffdshow. During the setup accept the defaults. (If MPEG1 and 2 are not selected, the setup program will automatically enable them in the registry.)

The driver is preinstalled with the setup program but may pop up a Windows Logo warning. Click "Continue Anyway". The driver will install without prompting for file locations.

If using Windows 2000, the setup program will check for DirectX9 and launch a browser on the Microsoft website if not present. DirectX9 is required on Windows 2000, but not XP. Please download the latest DirectX9 runtime from the Microsoft website if setup requests it.

Redistribution

The SDK CD contains the redistributables in the API directory. The following files are required for re-distribution:

- API\mid314.dll (installed in application exe directory)
- API\filters\ffmp2encoder.ax (needs registered with regsvr32)
- API\filters\m2tsmuxer.ax (needs registered with regsvr32)
- API\filters\smartdump.ax (needs registered with regsvr32)

- API\filters\wavdest.ax (needs registered with regsvr32)
- API\windir\s314param.ini (install to windows directory)

Additionally, ffdshow-20051221.exe should be redistributed for preview applications.

The drivers must also be redistributed to end-users and installed for proper function. They are included in the drivers directory.

SDK Reference

Release Notes

August 24, 2007

- Released source code for DLL (provided as is)
- Console demo cleaned up. Added callback demonstration to the console demo
- New callback functions add to SDK. Console Demo section added to manual.

V.1.0.1 (July 23,2006):

- Initial release

The common API flow is described below. Refer to the complete functions reference for the details on individual functions.

General Notes

The common API flow is described below. Refer to the complete functions reference for the details on individual functions.

1. Initialization. This is performed by a call to `SN_Open()` function. Initial default capture settings are loaded.
2. A call to `SN_Open()` may be optionally followed by calls to the functions controlling various settings:
    - input source: `SN_SetSource();`

- video preview window: SN_SetBoardWindow();
- video system and geometry: `SN_SetVidSys(), SN_SetVidSize();`
- video parameters (brightness, contrast, saturation, hue): `SN_SetLevels();`
- compression configuration: `SN_SetEncodeType(), SN_SetBitrate();`
- audio input configuration: `SN_SetAudioInput(), SN_SetMute();`
- record mode: `SN_SetRecordMode();`
- OSD: `SN_SetOverlayText()`.

3. A call to `SN_StartPreview()` starts the 314 for preview only. The stream received from the PCI bus is decoded and displayed in the user window specified with `SN_SetBoardWindow()`.

4. If recording is required, the stream should be started with a call to SN_StartRecord, and stopped with a call to `SN_StopRecord()`.

5. If Data needs to be retrieved from the Directshow Capture graph, please see the console demo and the functions SN_StartCaptureOnly, SN_StartCaptureOnly_CB and SN_StartCompressedCaptureOnly(mpeg callbacks).

6. During the recording the following function could be used to obtain some useful information:
   - `SN_GetStatus()` – current status (record, playback), current recorded file size and path;

7. `SN_Close()` must be called before application terminates to clean up properly.

Demo applications

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

app-314-demo – an MFC application. Displays the stream in the window, allows modification of compression, video, and audio parameters, recording the stream to the hard drive. More information about the MFC application is in the QuickStart manual on the website.

code_console – a simple console application. If you want to get the MPEG data or uncompressed data directly out of the 314 Card without preview, please see the console app source code.

Example console app usage

Compressed MPEG stream(cbmpeg)

The compressed MPEG stream in the console demo uses callbacks to get every MPEG frame(or JPEG for Motion JPEG) as it passes through the SampleGrabber. To start the VES(video elementary stream) of the compressed data, type "cbmpeg f" in the console demo where f is the filename. Currently, only encode modes MPEG1, MPEG2, and MPEG4 DIVX(encode=9) produce useable results. MotionJPEG capture is not demonstrated, but can be easily added by the user. MPEG4(Microsoft) encode=2 is NOT supported because this mode has no sequence headers.

MPEG1 VES Capture example commands

- "encode 0"
- "cbmpeg c:\\mpeg1_ves.mpg"
- wait 10 seconds
- "stop"

MPEG2 VES Capture example commands

- "encode 1"
- "cbmpeg c:\\mpeg2_ves.mpg"
- wait 10 seconds
- "stop"

MPEG4 VES Capture example commands

- "encode 9"
- "cbmpeg c:\\mpeg4_ves.mpg"
- wait 10 seconds
- "stop"

Multiplexed stream callbacks(cbmux)

MPEG1 PES Capture example commands

- "encode 0"
- "cbmux c:\\mpeg1_pes.mpg"
- wait 10 seconds
- "stop"

MPEG2 PES Capture example commands

- "encode 1"
- "cbmux c:\\mpeg1_pes.mpg"
- wait 10 seconds
- "stop"

MPEG4 and Motion JPEG are NOT supported for this mode. Please see SN_StartCompressedCaptureOnly function and the Regular Recording section below.

Split stream (separate Video and Audio callbacks)

The split stream commands allow separate video and audio callbacks. Please see the console demo.

Raw Uncompressed stream

"cbraw filename" starts an uncompressed capture stream using sample grabber callbacks. Inside the callback, a snapshot is saved each time with increasing frame index. Eg. The first snapshot is filename0.bmp, the second filename1.bmp, etc… This command uses the SN_StartCaptureOnly_CB to create the DirectShow filtergraph.

Regular Recording

The console application also demonstrate the standard recording functionality using SN_StartRecord. SN_StartRecord builds a full Capture Graph for recording. All encode types are supported. The AVI Mux rewrites the start of the MPEG4 and MotionJPEG AVI files when the stream is stopped. This is why MPEG4 and MotionJPEG are not supported by the "cbmux" callback multiplexing method. MPEG4DIVX is not supported by this function or "cbmux"

because a separate MPEG4 DIVX DirectShow multiplexer must be purchased and installed on the users system.

Functions Reference

All API functions are declared using the following definition:

#define MID314_API extern "C" __declspec(dllimport)

```
MID314_API int SN_Open( void );
```

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

Parameters

None.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_Close(
);
```

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

Parameters

None.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetBoardWindow(
    HWND hwnd,
    BOOL bResize,
    int board
);
```

Sets the window for display or rendering the video stream. If called, will automatically set SN_SetRenderVideo(TRUE).

hwnd

A handle to the window to display video in.

bResize

If set to TRUE, the window will be resized to match native video size.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetPlaybackWindow(
    HWND hwnd,
    BOOL bResize
);
```

Sets the window for playback.

hwnd

A handle to the window to display video in.

bResize

If set to TRUE, the window will be resized to match native video size.

```
MID314_API int SN_SetSource(
   MID314_SOURCE input,
   int board
);
```

Selects between composite and S-video inputs.

Parameters

input

enumerated input MID314_SOURCE (see mid314types.h). For model 314 the allowed values are MID314_SOURCE_COMPOSITE_0, MID314_SOURCE_COMPOSITE_1, MID314_SOURCE_COMPOSITE_2, MID314_SOURCE_COMPOSITE_3, MID314_SOURCE_SVIDEO_0 and MID314_SOURCE_SVIDEO_1.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_GetSource(
   MID314_SOURCE *pSource,
   int board
);
```

Retrieves current input settings.

Parameters

pSource

pointer to the value of current input.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetVidSys(
   MID314_VIDSYS vidsys,
   int board
);
```

Sets the input video system (NTSC, PAL).

Parameters

vidsys

video system enumerated type (see mid314types.h).

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetEncodeType(
   MID314_ENCODING encodeType,
   int board
);
```

Sets the desired encoding type (MPEG1,2,4).

Parameters

encodeType

encoding enumerated type (see mid314types.h).

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_GetEncodeType(
   int board
);
```

Retrieves current encoding type.

Parameters

board

board number in the system (use 0 for single board setups).

Returns

Current encoding setting, -1 if error.

```
MID314_API int SN_SetBitrate(
   int bitrate,
   int board
);
```

Set the desired bitrate of the output stream.

Parameters

bitrate

desired bitrate (in bits per second). Recommended values are 1,000,000 to 6,000,000 for D1 resolutions, 300,000 to 2,000,000 for CIF resolutions.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_GetBitrate(
   int board
);
```

Retrieves current bitrate setting.

Parameters

board

board number in the system (use 0 for single board setups).

Returns

current bitrate (in bits per second), -1 if error.

```
MID314_API int SN_GetStatus(
   MID314STATUS *pStatus,
   int board
);
```

Retrieves current status information (see mid314func.h for `MID314STATUS` type definition).

Parameters

pStatus

pointer to status variable.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_StartPreview(
   int board
);
```

Starts video and audio streaming for preview only.

Parameters

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_StopPreview(
   int board
);
```

Stops video and audio streaming for preview. If recording is enabled it is stopped as well.

Parameters

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetRecordMode(
   MID314_REC recMode,
   int board
);
```

Sets the record mode (see mid314types.h for `MID314_REC` type definition).

Parameters

recMode

record mode.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_StartRecord(
   char *fileName,
   int board
);
```

Starts recording to a file.

Parameters

fileName

full path to the target file, no extension.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_StopRecord(
   int board
);
```

Stops the recording. Preview will continue.

Parameters

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SnapshotToFile(
   char *fileName,
   int filetype,
   int qual,
   int board
);
```

Takes a snapshot and saves it to a file.

Parameters

fileName

full path to the target file, no extension.

filetype

file type(s) to save to (`MID314_FILE_JPEG`, `MID314_FILE_BMP`, see mid314types.h).

qual

JPEG quality (25-100, higher value yields better quality and bigger files).

board

board number in the system (use 0 for single board setups).

Returns

Image size in bytes or a negative value in case of an error.

```
MID314_API int SN_PlaybackVideo(
   char *fileName
);
```

Starts playback of the specified file in current window.

Parameters

fileName

full path to the target file.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_StopPlayback(
);
```

Stops video playback.

Parameters

none

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_PausePlayback(
   BOOL bPause
);
```

Pauses/resumes playback.

Parameters

bPause

TRUE for pause, FALSE for resume.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_PlaybackSetRate(
   double drate
);
```

Changes playback speed.

Parameters

drate

a double specifying playback speed. 0.5 corresponds to half of normal speed, 2.0 - double the normal speed. Minimum speed is 0.5.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_PlaybackSetSeekPosition(
   int percent,
   int range
);
```

Seeks the position in the file being played back. The position is calculated as file_size * percent / range.

Parameters

percent

a numerator of the position in the file expressed as a fraction of a total file size.

range

a denominator of the position in the file expressed as a fraction of a total file size.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_PlaybackGetSeekPosition(
   int range
);
```

Retrieves the current position in the file being played back. The position is calculated as file_size * percent / range.

Parameters

range

a denominator of the position in the file expressed as a fraction of a total file size.

Returns

a numerator of the position in the file expressed as a fraction of a total file size, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetLevels(
   int param,
   char value,
    int board
);
```

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

Parameters

param

defines the parameter to set (`MID314_LEVEL_CONTRAST, MID314_LEVEL_BRIGHTNESS, MID314_LEVEL_SATURATION, MID314_LEVEL_HUE`).

See see mid314types.h for definitions.

value

defines the value of selected parameter

(brightness 0 to 255 default 128,

saturation –128 to 127 default 64,

contrast –128 to 127 default 64,

hue –128 to 127 default 0).

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetMute(
   BOOL bMute,
    int board
);
```

Mutes audio on the host computer. Audio will still be recorded. If called while streaming, the function will stop and restart the sream.

Parameters

bMute

TRUE to mute the audio.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_GetMute(
   BOOL *bMute,
    int board
);
```

Retrieves current muting setting.

Parameters

bMute

pointer to retrieved setting.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetOverlayText(
   int idx,
   const POINT *pPos,
   const overlay_text_t *pOvlText,
   BOOL bEnable,
   BOOL bUpdate,
   int board
);
```

Configures text overlays (OSD).

Parameters

idx

Overlay number (index), 0-5.

pPos

position of top left corner of overlay window (in pixels). Position is rounded to the nearest 16x16 pixel block.

pOvlText

text to display (a maximum of 96 characters total for all overlay windows). See mid314types.h for the definition of `overlay_text_t`.

bEnable

TRUE enables current overlay region.

bUpdate

if setting multiple regions has to be FALSE for all regions except the last.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, -1 on failure, -2 – out of text space.

```
MID314_API int SN_ClearOverlay(
    int board
);
```

Clears all overlays.

Parameters

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetVidSize(
   int iVidSize,
   int board
);
```

Sets the captured video size (resolution).

Parameters

iVidSize

video size: 0 – full resolution, default (720x480 NTSC, 720x576 PAL), 1 – CIF (360x240 NTSC, 360x288 PAL).

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetRenderVideo(
   BOOL bDisplayVideo,
    int board
);
```

Turns the preview on and off. Recording will continue regardless of this setting.

Parameters

bDisplayVideo

TRUE to enable preview.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_GetRenderVideo(
   BOOL *bDisplayVideo,
   int board
);
```

Retrieves the preview setting.

Parameters

bDisplayVideo

pointer to retrieved setting.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetAspectRatio(
   MID314_ASPECT_MODE mode,
   int board
);
```

Allows modifying the video aspect ratio for preview.

Parameters

mode

MID314_ASPECT_NONE allows stretched preview image, MID314_ASPECT_CONST maintains original aspect ratio (see mid314types.h).

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_GetAspectRatio(
   MID314_ASPECT_MODE *mode,
   int board
);
```

Retrieves the aspect ratio control setting.

Parameters

mode

pointer to retrieved setting.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_Repaint(
   HDC hdc
);
```

Repaint callback, should be called when application receives a WM_PAINT event. This is necessary to notify the video renderer of a repaint event.

Parameters

hdc

device context handle or NULL if unknown.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_DisplayChange(
```

```
);
```

Informs the video renderer that a WM_DISPLAYCHANGE message has been received by the application. This callback should be called when the application has a WM_DISPLAYCHANGE event.

Parameters

none

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetBoardWindow(
   HWND hwnd,
   BOOL reserved,
   int board,
);
```

Sets the window for video streaming. Overrides SN_SetRenderVideo setting to TRUE.

Parameters

hwnd

Handle to video display window.

reserved

reserved parameter.

board

current board number.

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_SetRemoveMsg(
   HWND hwnd,
   int removeMsg,
   int board
);
```

Sets the value of the message that is sent to the application's window in case of device removal (e.g. USB cable unplugged).

Parameters

hwnd

handle of the window that will receive the message.

removeMsg

message value to be sent.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_TestDeviceRemoval(
    int board
);
```

Tests whether device was removed or not. When `SN_SetRemoveMessage` is called, a message window handle and remove message is set. When remove message is received by the application, this function is called to confirm the device was removed and if it was, the DLL should be shut down.

Parameters

board

board number in the system (use 0 for single board setups).

Returns

0 in case the device was not removed, 1 in case it was.

```
MID314_API int SN_SetVideoPosition(
   int xpos,
   int ypos,
   int xsize,
   int ysize,
   int left_clip,
   int top_clip,
   int right_clip,
```

```
   int bottom_clip,
   int board
);
```

Sets the video position in the clipping window.

Parameters

xpos, ypos

x and y coordinates of the top left corner.

xsize, ysize

horizontal and vertical sizes of video display.

left_clip, top_clip, right_clip, bottom_clip

number of pixels to clip of the left, top, right and bottom sides of source video.

board

board number in the system (use 0 for single board setups).

Returns

0 on success, negative value if error (see mid314types.h for error codes list).

```
MID314_API int SN_GetNumBoards(
   int *pNumBoards
);
```

Retrieve the number of boards in the system.

Parameters

pNumBoards

Returns the number of boards.

Returns

0 if success, negative if error.

```
MID314_API int SN_GetFrame(
   long inSize,
   unsigned char *pFrame,
   BITMAPINFOHEADER *pBMI,
```

```
   int type,
   int board
);
```

SN_GetFrame retrieves current frame. Type Currently supported for MPEG 1,2,4 only. Please see GetFrame.cpp in demo app for an example of using SN_GetFrame. The function is **not** available for the callback filter graphs (SN_StartCompressedCaptureOnly and SN_StartCaptureOnly_CB)

Parameters

inSize

size of pFrame buffer supplied (should be at least 768*576*3)

pFrame

pointer to buffer of size inSize where frame is stored

pBMI

pointer to bitmap info. Currently only the following parameters are updated: pBMI->biHeight, biWidth, biSizeImage.

type

type of buffer to retrieve.

type = 0 for YcbCr buffer formatted as follows. Y plane first 768*576 bytes. Cb plane next 768*576/4 bytes. Cr plane next 768*576/4 bytes.

type = 1 for RGB8 buffer.

board

Board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_SetMD(
   MID314_MDConfig_t *pMD,
   HANDLE event,
   int board
);
```

Sets or updates the motion detection region according to MID314_MDConfig_t (see mid314types.h file for description). The main demo app shows an example of using motion detection.

Parameters

**pMD*

pointer to MD structure

event

handle to event to notify.

board

Board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_StopMD(
   int board
);
```

Stops motion detection.

Parameters

board

Board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_DisplayMDRegions(
   BOOL bOnOff,
   int board
);
```

Displays motion detection regions on displayed image (if rendering the video to screen). Requires Windows XP or DirectX 9 (Windows 2000).

Parameters

bOnOff

display on or off

board

Board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_GetMDData(
   int *region,
   unsigned char *buf,
   int bufsize,
   int board
);
```

Retrieves the motion detection data. See demo application for example.

Parameters

region

the returned region mask. 0x02: region 1, 0x04: region 2, 0x08 region 3. The region mask should be used to detect if motion detected or not.

buf

a pointer to motion macroblock map (must be size 208).

bufsize

size of bufmap. MUST use 208.

Board

board number in system.

The primary detection is by region. If further information is wanted, the MB map will show if motion passed threshold in a given region on a macroblock by macroblock level (16 pixels x 16 pixels) on a bit level. For example, (buf[0] & 0x01) represents macroblock 1, (buf[0] & 0x02) represents macroblock 2, (buf[1] & 0x01) represents macroblock 8.

Returns

0 if success, negative if error.

```
MID314_API int SN_StartCaptureOnly(
   int colorspace,
   int board
);
```

This function builds a DirectShow Capture graph with the Raw Capture pin connected to the DirectShow SampleGrabber. The Sample Grabber runs in buffered mode. This mode is useful for occasional snapshots. If all snapshots are required, the user should use SN_StartCaptureOnly_CB, which is the same capture graph, but uses callbacks. In this mode, there is no preview available. Motion Detection is not supported with this filter graph.

Parameters

colorspace

The desired colorspace for the SampleGrabber. 0-RGB24, 1-YUY2.

board

board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_StopCaptureOnly(
   int board
);
```

Stops the graph started by SN_StartCaptureOnly.

Parameters

board

board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_StartCaptureOnly_CB(
   int recmode,
   int colorspace,
   int board
);
```

This function is used to get uncompressed video data out of the directshow graph. The function builds a DirectShow Capture graph with the Raw Capture pin connected to the DirectShow SampleGrabber. The Sample Grabber runs in Callback mode. The callback function must not block. This mode is useful for directly getting all samples without polling SN_GetFrame. Also, SN_GetFrame can not detect when a new frame occurs because of the nature of the SampleGrabber buffering.

The recording mode, *recmode*, determines how the DirectShow graph is built. The function should be used if no preview is required and the user just wants to get uncompressed frames or uncompressed frames with audio from the 314.

There are three modes supported currently. “recmode=0” is a filter graph with the 314 Uncompressed Capture pin connected to the SampleGrabber. The SampleGrabber filter is then connected to a Null Renderer as required by the DirectShow SDK. The callback function is called when uncompressed frames are received. It MUST be registered with SN_RegisterVidCB before calling this function. Otherwise, the graph gets built and no frames are delivered.

“recmode=1” and “recmode=2” are the same as “recmode=0” except the audio is also capture. “recmode=1” has encoded audio and “recmode=2” has raw audio. The audio callback MUST be registered with SN_RegisterAudCB before calling this function.
There is no preview supported in this callback filter graph.
Therefor, any preview related functions do not apply(SN_SetBoardWindow, SN_SetRenderVideo, etc..)

Parameters

recmode

Recording mode(see above)

colorspace

The desired colorspace for the SampleGrabber. 0-RGB24, 1-YUY2.

board

board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_StopCaptureOnly_CB(
   int board
);
```

Stops the graph started by SN_StartCaptureOnly_CB.

Parameters

board

board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_StartCompressedCaptureOnly(
   int recmode,
   int board
);
```

This function is used to get compressed video data and/or audio data out of the directshow graph. The function builds a DirectShow Capture graph with the MPEG Capture pin connected to the DirectShow SampleGrabber. The Sample Grabber runs in Callback mode. The callback function must not block. The recording mode, *recmode*, determines how the DirectShow graph is built. The function should be used if no preview is required and the user just wants to get compressed frames or compressed frames with audio from the 314.

There are three modes supported currently. “recmode=0” is a filter graph with the 314 compressed capture pin connected to the SampleGrabber. The SampleGrabber filter is then connected to a Null Renderer as required by the DirectShow SDK. The callback function is called when compressed frames are received. The callback function MUST be registered with SN_RegisterVidCB before calling this function. Otherwise, the graph gets built and no frames are delivered.

“recmode=1” puts a SampleGrabber at the output of the multiplexer of the compressed video and encoded audio data. The console demo demonstrates the use of this mode. Currently only encode types of MPEG1 and MPEG2 are supported with this mode. The AVI Mux used for MP4S MPEG4 or Motion JPEG does not support streaming so data can’t be saved using callbacks. MPEG4 data and MotionJpeg data is available, but modes 0, 3 or 4 should be used for them.

“recmode=2” is the same as “recmode=1” but no audio is added to the multiplexer.

“recmode=3” and “recmode=4” are the same as “recmode=0”, but have an additional audio stream. “recmode=3” has encoded audio and “recmode=4” has raw audio. The audio callback MUST be registered with SN_RegisterAudCB before calling this function to get the audio data. This is shown in the console demo.

There is no preview supported in this callback filter graph. Therefor, any preview related functions do not apply(SN_SetBoardWindow, SN_SetRenderVideo, etc..)

Parameters

recmode

Recording mode(see above)

board

board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_StopCompressedCapture(
   int board
);
```

Stops the graph started by SN_StartCompressedCapture.

Parameters

board

board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_RegisterVidCB(
  Cbfunc_t callback_func,
  int board,
  );
```

Registers a callback function for use with the SampleGrabber DirectShow graphs. Used with SN_StartCompressedCaptureOnly and SN_StartCaptureOnly_CB. Please see the console demo for a full example.

Parameters

callback_func

The callback function for video(RAW, compressed, or muxed PES(includes audio))

board

board number in system.

Returns

0 if success, negative if error.

```
MID314_API int SN_RegisterAudCB(
  Cbfunc_t callback_func,
```

```
int board,
);
```

```
Registers a callback function for use with the SampleGrabber
DirectShow graphs.  Used with SN_StartCompressedCaptureOnly(recmodes 3
and 4) and SN_StartCaptureOnly_CB(recmodes 1 and 2). Please see the
console_demo for a full example.
```

Parameters

callback_func

The callback function for audio (encoded MP2 or raw)

board

board number in system.

Motion Detection

Motion Detection on the 314(and 2250) involves regions of interest and motion thresholds. The chip firmware has 3 user programmable regions of interest. The SDK shows 4 regions of interest in the structure MID314_MDConfig_t. The 4th region, corresponding to index 3(starts from zero) is reserved for possible future use.

The regions are described in pixels. The user specifies an upper left co-ordinate and a lower right pixel coordinate for each region.

There is a requirement that the regions do not overlap on a macro-block(MB) level. They should be spaced 16 pixels apart as one macro-block corresponds to a 16x16 pixel block. The MDSettingDlg.cpp file in the demo app demonstrates verification of the MD regions.

If a region is unused, the coordinates {0,0} should be used for both the upper left corner and bottom right corner.

In each region, motion is detected on the basis of threshold values. The first threshold is the SAD (sum of absolute differences), which represents changes in intensity. The recommended range for the SAD is 20-150. In general SAD will catch fast motion across the target. For testing purposes, start at SAD=40 (with sensitivity = 80).

The second threshold is the motion vector threshold. The recommended range is 0-1600. For testing purposes(with sensitivity = 80), start at MV=700. The MV threshold will detect slow motion across the target better than SAD.

Finally, there is a sensitivity value. It may be set from 0-100. A recommended start setting is 80. In general, higher thresholds may be used for higher sensitivity settings.

Mathematically, motion detection in the chip is calculated as follows:

H/w comes up with a set of MV for each MB in each direction, MV = ((Vx * Vx) + Vy*Vy)) >> 2).

If ((Vx*Vx+Vy*Vy)>>2)>MV_THRESHOLD), then this MB is motion detected.

If (Sad>SAD_THRESHOLD), then this MB is also motion detected.

If the detected MB is in a specified region, a motion map is produced and a notification sent to the user.

The motion detection configuration structure is as follows (mid314types.h) (The last region is reserved):

```
typedef struct
{
    MID314_PointCorrds_t ULPoint[ 4 ];
    MID314_PointCorrds_t BRPoint[ 4 ];
    unsigned short u32SADThresholdValues[ 4 ];
    unsigned short u32MVThresholdValues[ 4 ];
    unsigned short u32SensitivityValues[ 4 ];
} MID314_MDConfig_t;
```

Motion Map

When motion is detected, an event is generated with data containing the region(s) of interest with motion and a motion map. The motion map contains all the MBs which were hit inside a given region. Note that some MBs will be shown outside the region (please see Example 2 in the motion examples below). If the user sets one region to cover the whole image, then the Motion Map will represent motion hits across the whole image. This allows the user to implement a custom MD method on a macro-block by macro-block level.

A sample motion map where the region covers the full screen is shown below. The blank lines on the bottom are not part of the image because the image was NTSC(480) not PAL(576). The demo map shows macro-blocks up to number 36(576/16) in the vertical direction. The motion map (and hit statistics) can be viewed by clicking "Show Map".

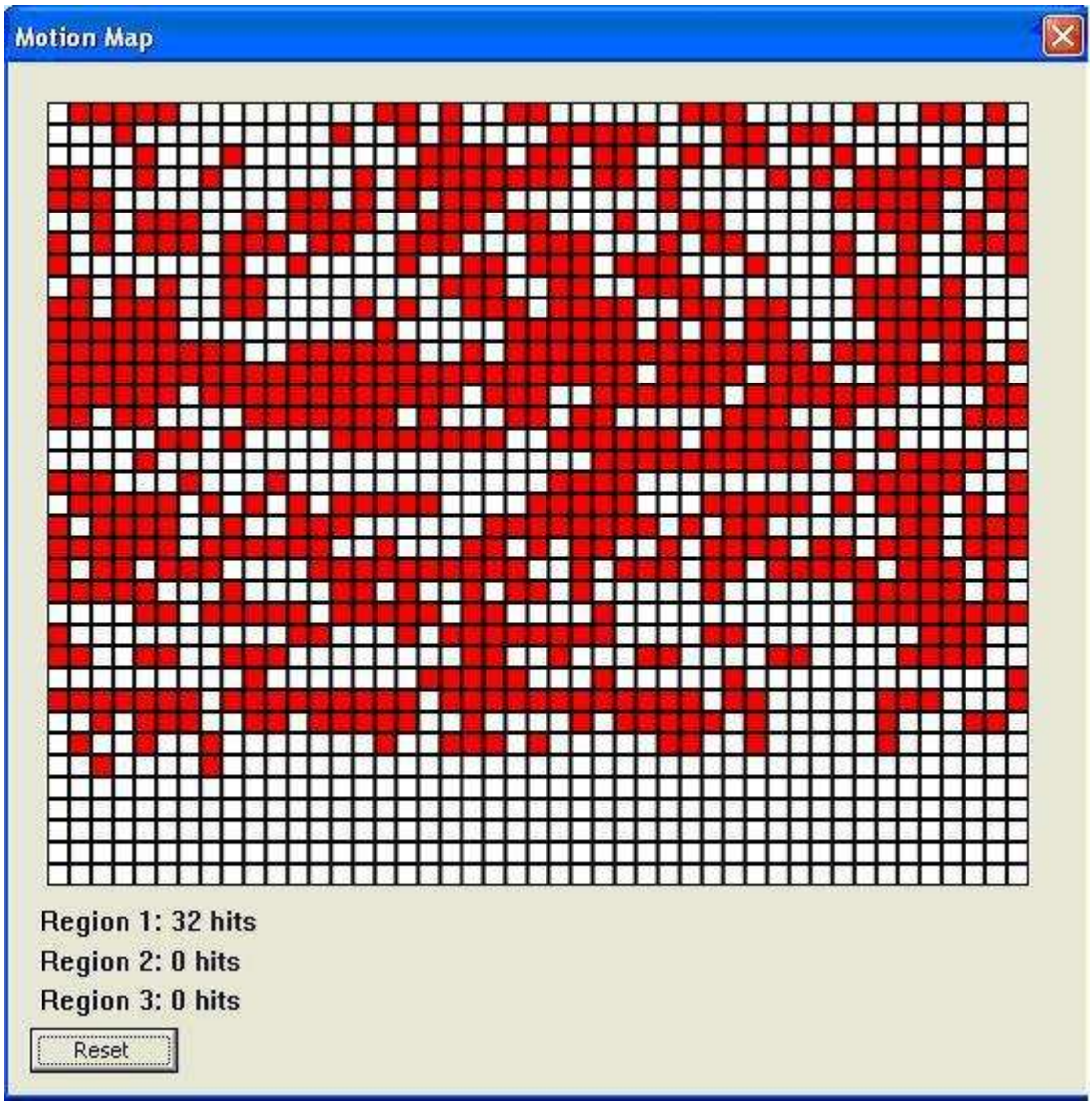

Figure 1 Motion Map(statistics represent number of motion events per region).

The motion-map above shows a hit at macro-blocks 2,3,4,5 and other macro-blocks where the squares are red. The white color MB #1 does not have a hit in this motion map. Macroblock 2 = pixels (16,0) to (31,16). The 32 hits corresponds to 32 motion events received by region 1. It does **not** represent the total number of red macroblocks(motion macroblocks) in the motion map.

Motion Detection Quickstart

The demo application shows an example of motion detection. A step by step example is shown below:

- Select Configure in the motion detection group box(lower right corner of dialog).
- Select a region or regions. Click on Enable Region1 Setting.

- Enter 720 for BR_x and 320 for BR_y.
- Click OK.
- Start stream (if not already started).
- Select "Show Regions". Region selected above will be shown on the image.
- Select "Show Map". A macro-block map with detailed hits for region 1 above will be shown along with a counter of the number of hits in region 1.
- To disable motion detection, call SN_StopMD or in the demo app, disable all MD regions.

Motion Map Example

Example 1: Three regions of interest

The following motion detection regions were defined. An overlay of the defined regions can be seen by selecting the Configure button in the dialog window.

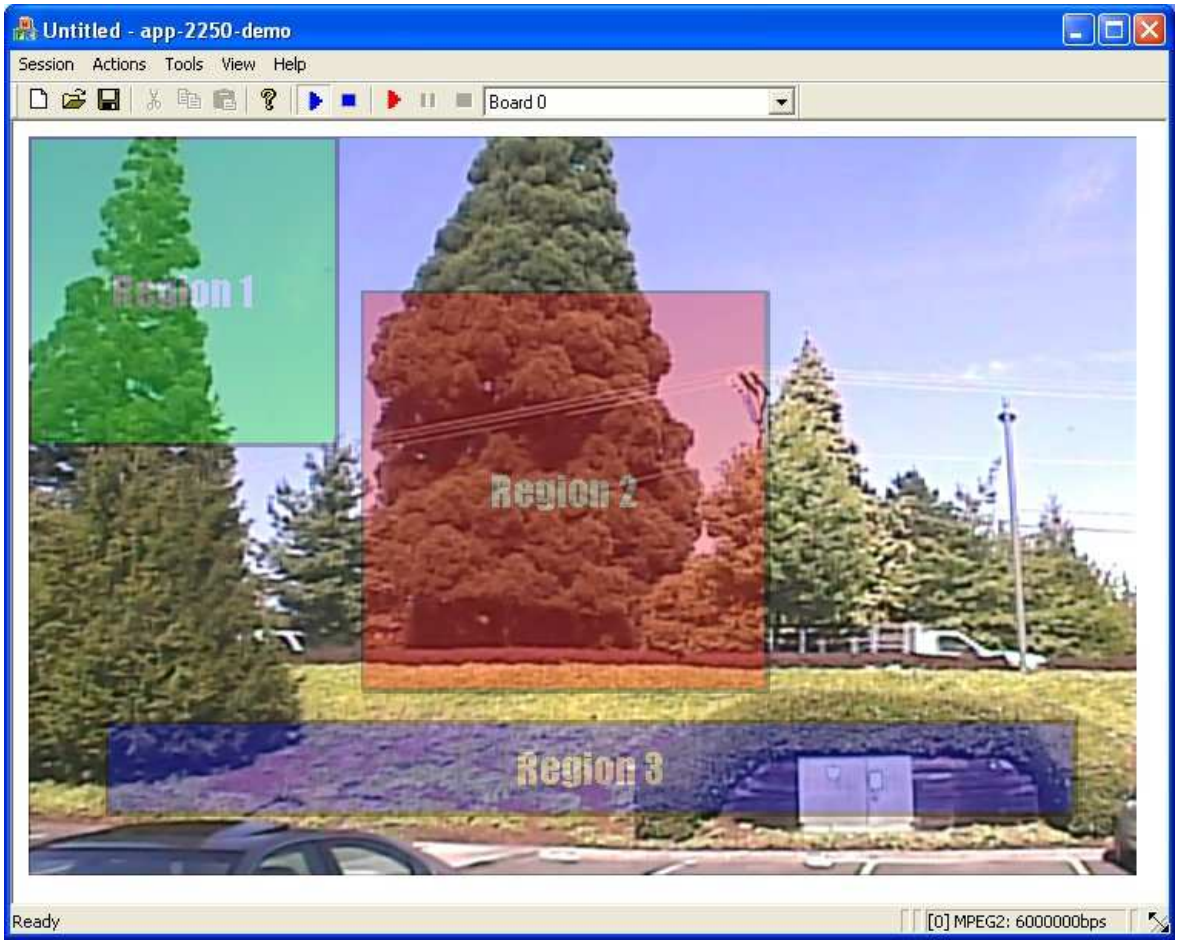

Figure 2 : Three(3) Motion Detection Regions of Interest

The above regions were set-up by selecting the "Configure" button. A screen shot of the setup for the regions above is show in figure 3.

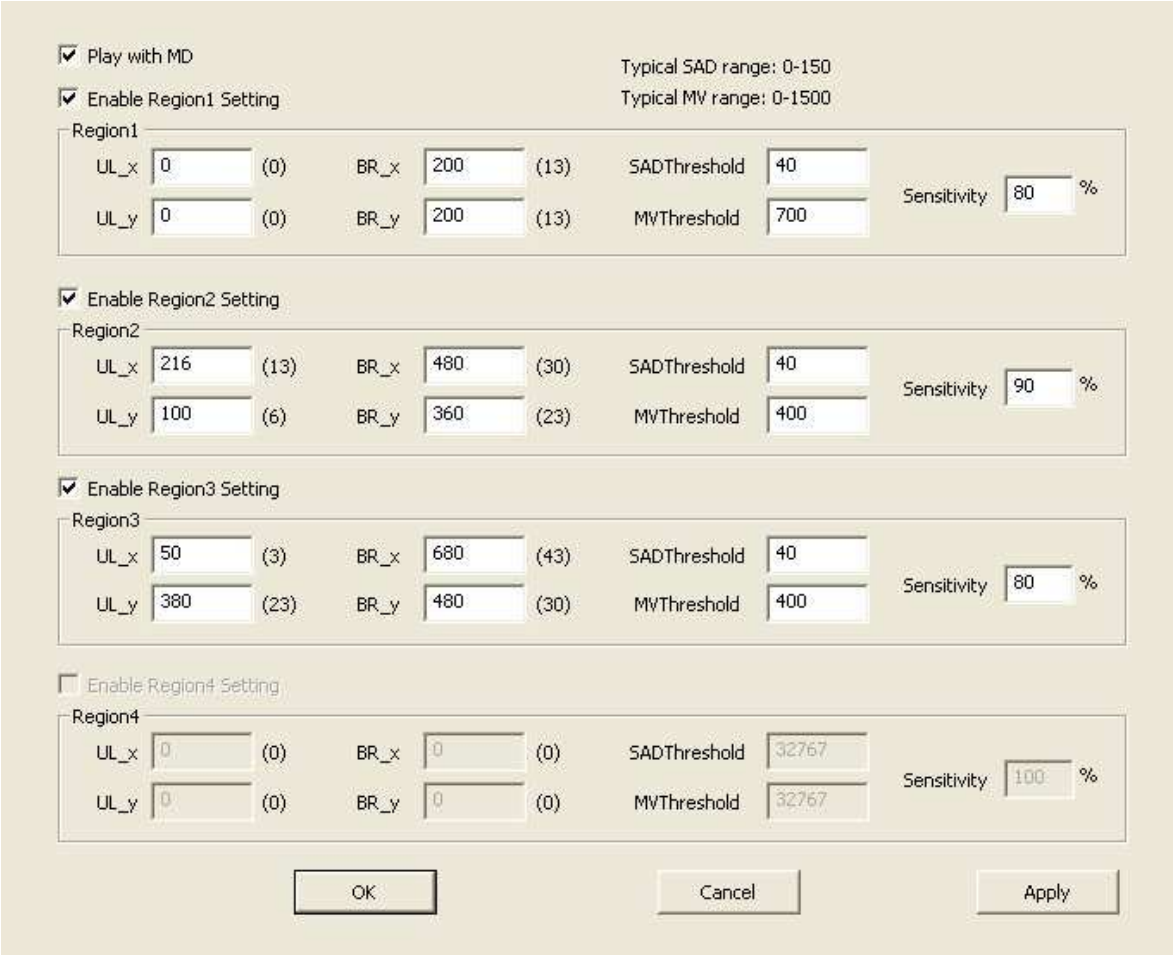

Figure 3: Setup for regions shown in Figure 2

Example 2 : Advanced Motion Map

The motion map may show macroblocks outside the range of interest. This example demonstrates how the motion map works. In short, only those macroblocks inside the defined areas of interest will show motion. Some macroblocks outside the region of interest may show motion. These should be ignored.

In this example, a small motion detec tion region in the upper left corner is produced as shown in Figure 4.

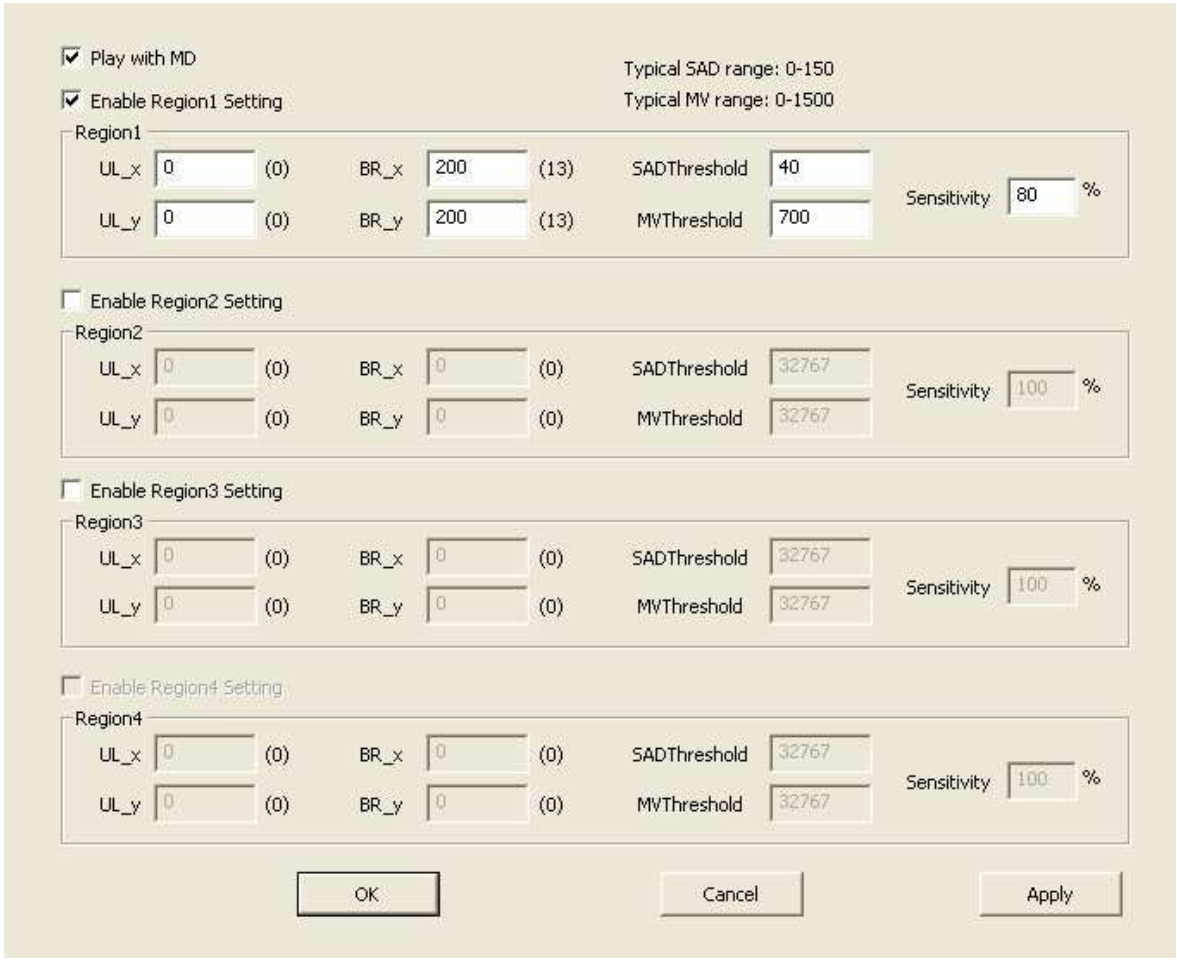

Figure 4

This corresponds to the region show in Figure 5.

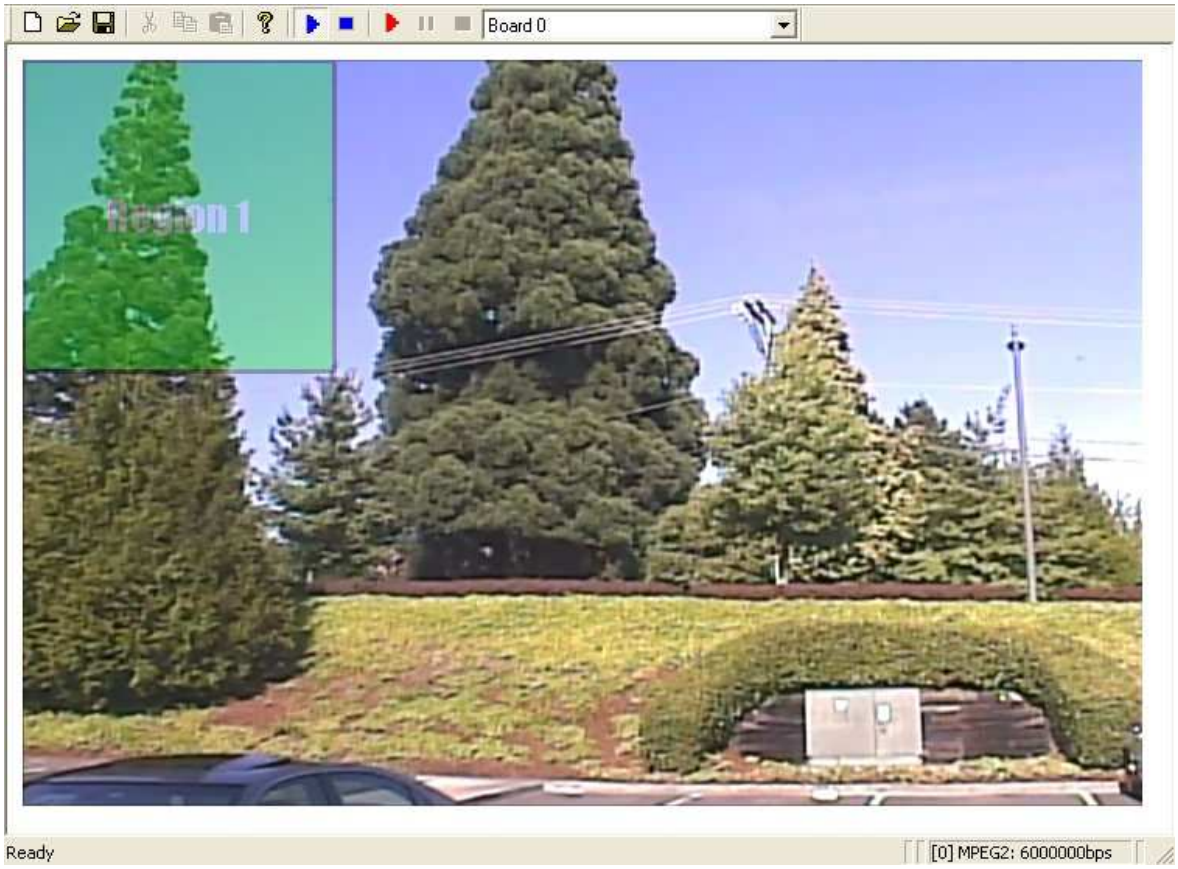

Figure 5: Small region in upper left corner

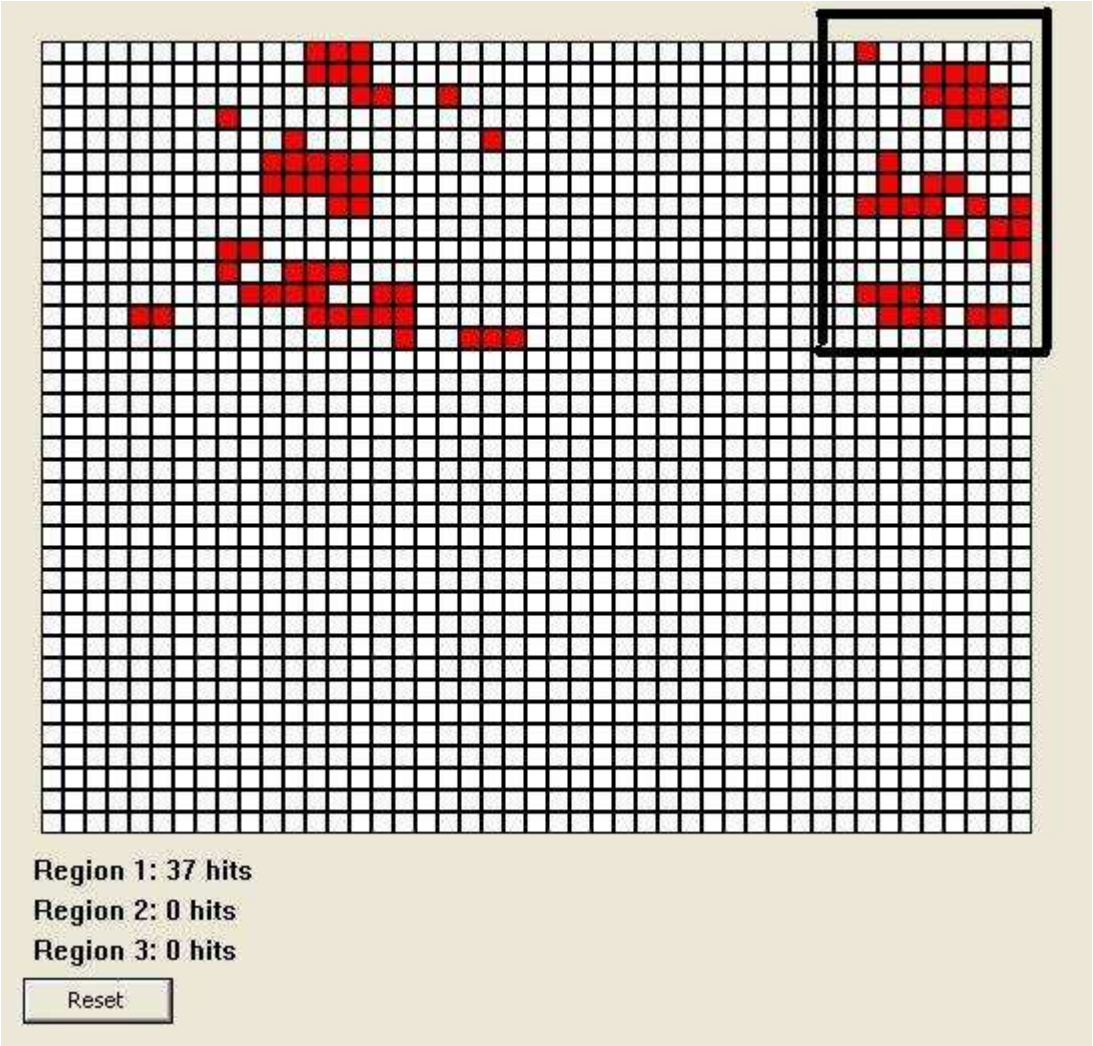

Figure 6

The produced motion map when motion occurs is shown in Figure 6. The black rectangle shows macroblocks outside the area of interest. As example earlier, these may be ignored. The number of hits 37 is the number of times a motion event occurred in region 1. It is not a count of the macroblocks in the region.

Advanced: Note that no macroblocks are shown below the region in the y direction. When a motion map is created, it scans from left to right starting at the top and going to the bottom. Any macroblock motion along the scan line is included in the motion map. So stray motion events(which may be ignored) occur beside the region, but not above or below the region.

Example 3: (Advanced) Custom motion detection

Some customers may want motion information on a macroblock by macroblock basis. If this is the case, then only one region should be defined. This region should encompass the whole picture in order to get all the macroblock events included in the motion map.

Step 1:

Set up the region to cover the full screen as shown in Figure 7.

Figure 7

Step 2:

Process the motion map. Figure 8 shows a processed image map from the returned event data.

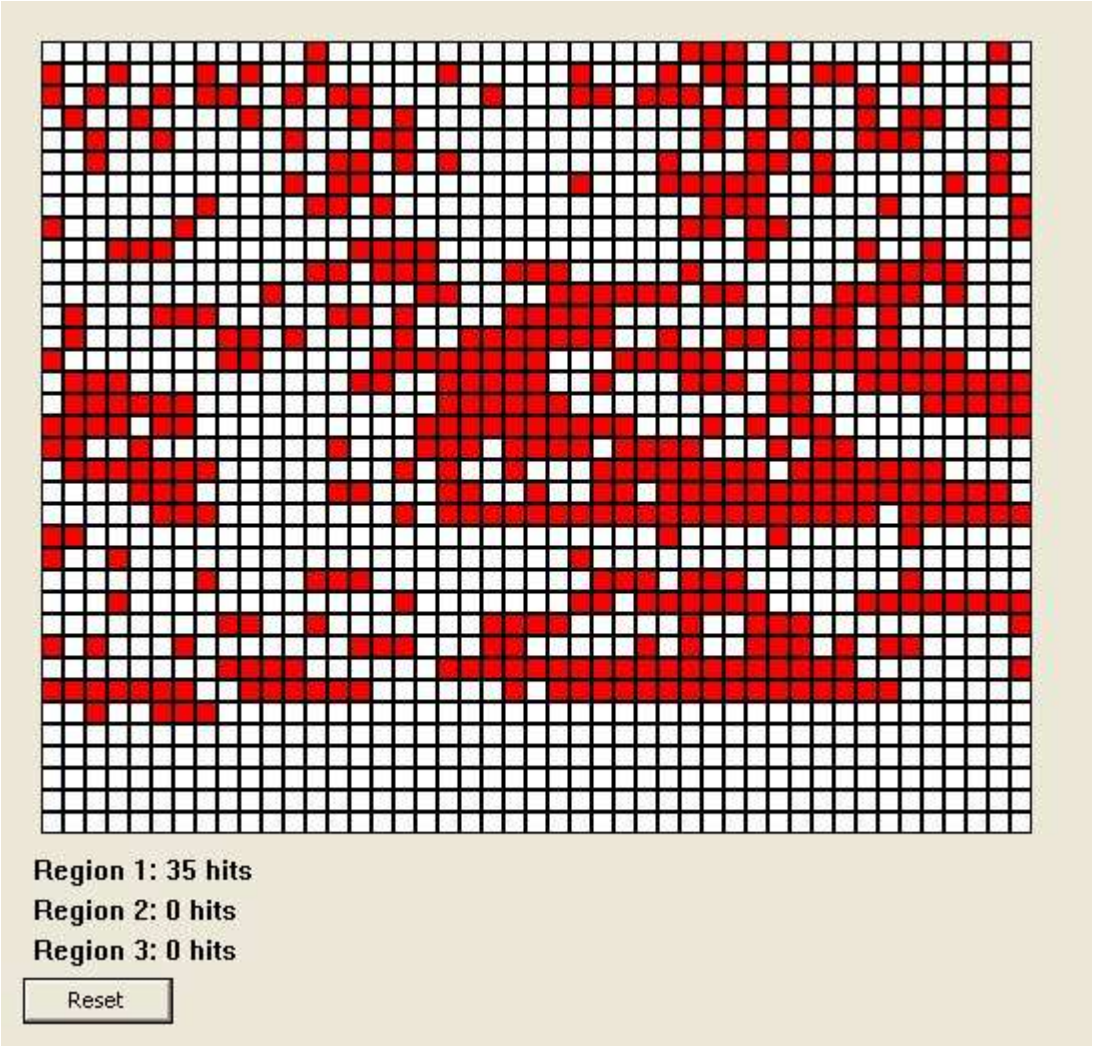

Figure 8

When the event is captured by the waiting user program(see demo source code), it calls SN_GetMDData(&region, pBuf, BUFSIZE, board_number). The region(s) where motion was detected is given in a bitmask in the region variable. BUFSIZE should be set to 208, the size of the motion map. pBuf will contain the macroblock map on a bit by bit basis.

For example, assume the following is returned by SN_GetMDData where pBuf is an array of unsigned 8 bit characters.

pBuf[0] = 0x7c;

pBuf[1] = 0x00;

pBuf[2] = 0x89;

From pBuf[0], we have the motion detect events for the first 8 macroblocks. 0x7c= 01111100b which represents the following:

Macroblock 1 : bit field = 0, so no motion.

Macroblocks 2-6 : bit field = 1, so motion detected

Macroblocks 7-8: no motion detected.

pBuf[1] = 0x00, so macroblocks 9-16 have no motion in them.

The motion map given in pBuf thus allows a user to create a custom motion detection algorithm(The threshold will be the same for each macroblock) where they can generate motion detection on an individual macroblock level (16x16 pixels).